STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 • U.S.A.

March 1975

Annual Report

RESEARCH IN INTERACTIVE SCENE ANALYSIS

DRA

By: Jay M. Tenenbaum, Thomas D. Garvey,
Stephen Weyl, Helen C. Wolf

Prepared for:

National Aeronautics and Space Administration
Headquarters
Room 607
Washington, D.C. 20546

Attention: Charles Pontious

CONTRACT NASW-2086

SRI Project 8721

Approved by:

Bertram Raphael, Director
Artificial Intelligence Center

Bonnar Cox, Executive Director
Information Science and Engineering Division

ABSTRACT

An Interactive Scene Interpretation System (ISIS) is being developed by Stanford Research Institute's Artificial Intelligence Center as a tool for constructing and experimenting with man-machine and automatic scene analysis methods tailored for particular image domains. This report describes a recently developed region analysis subsystem based on the paradigm of Brice and Fennema. Using this subsystem a series of experiments was conducted to determine good criteria for initially partitioning a scene into atomic regions and for merging these regions into a final partition of the scene along object boundaries. Semantic (problem-dependent) knowledge is essential for complete, correct partitions of complex real-world scenes. An interactive approach to semantic scene segmentation was developed and demonstrated on both landscape and indoor scenes. This approach provides a reasonable methodology for segmenting scenes that cannot be processed completely automatically at present, and is a promising basis for a future automatic system. The report also describes a program that can automatically generate strategies for finding specific objects in a scene based on manually designated pictorial examples.

CONTENTS

FIGURE CAPTIONS

TABLES

ACKNOWLEDGEMENTS

We are grateful to R. Ohlander and D. R. Reddy of Carnegie Mellon University for supplying us with accurately digitized images.

1. INTRODUCTION

This report describes recent research in interactive scene analysis using the Interactive Scene Interpretation System (ISIS) developed at SRI. Our overall objectives are to: (1) devise interactive methodologies for rapidly programming computers to recognize objects in particular real-world pictorial domains, and (2) develop techniques for cooperative scene anlysis whereby humans can provide the computer with guidance when completely automated processing is infeasible.

Scene analysis is still more of an art than a science. The field is characterized by a few general principles and a growing collection of ad hoc techniques perfected, largely by trial and error, for particular domains. Even experienced researchers have difficulty predicting which, if any, of these techniques apply to a new domain. ISIS was originally developed to help researchers refine their intuitions. Data can be observed on a gray scale or color display as it is perceived by the computer. Primitive operators can be applied to selected areas of the scene to determine directly (in numerical terms) which features provide the best discrimination among particular objects. Object finding strategies can be formulated in terms of these features and tested on-line, by requesting the system to illuminate all instances observed in a displayed scene. It was possible to construct strategies for finding each of the principal surfaces in a simple room scene (e.g., floor, tabletop, chairseat, and wall) in a matter of hours [1]. These strategies were based on primitive operators for extracting hue, saturation, height, and

surface orientation from correlated arrays of color and simulated range data. Attribute descriptions (e.g., the color buff and the orientation horizontal) were developed in a similar fashion and used to express additional object finding strategies in symbolic terms.

ISIS is a step toward more natural pictorial communication with machines. Ideally, one would like to program a computer to find unfamiliar objects as one would instruct a person by pointing at examples, or by providing crude verbal descriptions. A tree, for example, might be described as a "green, leafy region above a tall, brown, vertical, bark-textured region." Unfamiliar concepts such as "green" or "leafy" could in turn be defined by pointing at examples. The computer might demonstrate comprehension by outlining instances of the described object in a displayed image. The programmer could then refine the description empirically to correct errors in the computer's interpretation. While we are still far from achieving this ideal, some significant progress has been made.

1.1 Summary of Research Progress

This project has addressed several major limitations of the initial version of ISIS described in Reference 1. First, the preliminary system lacked display capabilities, primitives, and scene segmentation techniques needed to function effectively in natural (i.e., outdoor) scenes. Second, while providing helpful tools, it left the major programming burden with the user. Third, the system provided no capabilities for high-level graphical communication such

as "pointing."

Our ability to observe data as they appear to the computer was significantly enhanced by the acquisition of a color image display. The display was particularly helpful in investigating color and texture discrimination in landscape scenes. Unlike painted objects found in room scenes, most naturally occurring objects cannot be distinguished solely on the basis of locally sampled attributes such as hue and saturation. Therefore, outdoor scenes must be partitioned into coherent regions so that global attributes (e.g., texture and shape) and spatial relations with other regions (e.g., adjacency, vertical position, and the like) can be used.

Various syntactic and semantic techniques for scene partitioning were experimentally evaluated using ISIS. A new interactive approach to semantic scene segmentation was demonstrated on both landscape and indoor scenes. This approach is of interest both as a reasonable way to segment scenes that at present cannot be processed completely automatically and as a promising base for a future automatic system.

Progress was made in automating the interactive generation of strategies for finding objects in room scenes. Specifically, a program was developed which accepts pictorial examples of object surfaces (e.g., floor, wall, tabletop, and other objects) outlined on a display screen by a human trainer, and attempts to design a good strategy for selecting samples of those

surfaces in future images. The program emulates the empirical approach of a human trainer: the example is first characterized in terms of feature extraction operators. A set of features is selected that distinguishes the example from examples of previously learned concepts. Finally, a corresponding predicate is generated and applied to random image sample points. The selected feature set is empirically refined, if necessary, to exclude sample points not belonging to the example, or to include omitted samples of the example. The human trainer can guide the machine's experimentation by describing the new object in terms of attributes of previously characterized objects or by directly suggesting which operators to try.

Designating examples by pointing with a cursor is usually more natural than drawing a detailed outline, especially for spatially amorphous objects such as trees. However, pointing is intrinsically ambiguous: the trainer could be designating anything from the particular picture element to the entire picture. The machine must first guess the example that it is expected to describe. Two pointing inference mechanisms were devised, one based on region growing for use in outdoor scenes and the other on automatic strategy generation for use in indoor scenes. In the first instance, the region grower was modified to grow outward from designated starting points inside and outside the object of interest. In the second case, a strategy was generated specifically for distinguishing between samples designated to be on an object of interest and those designated to be on adjacent objects. In both cases, after the computer's guess is displayed, the trainer can elaborate on his

intent by designating additional samples on and off the object. The guessing process then iterates until trainer and computer agree.

The interactive techniques described in this report are not yet integrated into a single system. Moreover, many of the scene analysis results have yet to be rigorously tested in a large number of scenes. The project has recently reached a very fertile experimental stage, as indicated by the fact that new ideas for scene analysis are occurring much more frequently than ideas for improving the system.

1.2 Development of QLISP Language

A portion of this project has supported development of a high-level programming language called QLISP. The QLISP language provides most of the features of QA4 embedded within the INTERLISP system. Thus, QLISP provides a rich variety of data types, a data base for content-directed retrieval of expressions, a powerful pattern matching capability, pattern-directed function invocation, and a mechanism for manipulating data contexts. These are all available in a programming environment that provides a versatile, LISP-oriented editor; an easy-to-use file package for maintaining symbolic files; a "programmer's assistant" which allows commands to be undone or altered; and an error correction system which can fix many simple user errors automatically.

Although QLISP has not been used as the programming language for the work described elsewhere in this report, the

requirements of that work have provided direction to the effort to streamline a sophisticated but costly language into an efficient tool for problem solving.

During the period covered by this report, the basic features of the language were implemented and incorporated into a rather reliable and easy-to-use package. A compiler has been implemented, and other modifications have been made to improve the language's efficiency. Compiled programs written in QLISP run about 30 to 50 times faster than the corresponding QA4 programs.

The most serious drawback to QLISP is that it tends to force the programmer who uses it into an unnecessarily restricted framework. Thus, for example, QLISP's data storage and retrieval statements are implemented in such a way as to prohibit the use of the data base as a network of items that could be easily traversed. Furthermore, the "grain" of the available statements is sometimes too coarse; i.e., it is not always possible to specify precisely how a retrieval operation should be carried out. These are the primary reasons why it was decided not to implement ISIS in QLISP.

However, QLISP has been shown to be a useful tool for problem solving, simulation, and automatic programming. Therefore, work will continue, under other support, to further improve and streamline this language. During the coming year, we plan to install a new pattern matcher which can perform unification (matching two patterns, both of which contain variables) as well as do simple matches more quickly. We will install a new data base

mechanism that will allow more efficient access of related structures in the associative store. We will investigate the extension of the associative store to secondary storage. Finally, when the Bobrow-Wegbreit control stack formalism is incorporated into INTERLISP, we will add facilities for pseudo-parallel processing to QLISP.

QLISP is available over the ARPA net, and has already been used on an experimental basis at several sites around the network.

2. SYSTEM DEVELOPMENT

A substantial amount of effort during the past year was devoted to system development. A simple digitizing device for photographic transparencies was constructed. A color display was acquired and interfaced, necessitating the development of both low- and high-level color display software. The original ISIS system described in Reference 1 was extensively overhauled to improve both efficiency and memory utilization.

2.1 Hardware

2.1.1 Slide Digitizer

A direct view slide digitizer was constructed using a sheet of translucent Polakote as the optical interface. Images viewed through a ground glass interface had undesirable grain, and images viewed on a conventional projection screen had dull, unsaturated color. A 1:1 field lens was also tried as a direct view interface. It provided brilliant colors but the image was too small for the zoom optics built into our camera.

Pictures were digitized at a sampling resolution of 120 X 120 elements, 5 bits/color. The camera gain and iris were peaked for each color filter, so that the brightest white in the scene (usually a cloud) was barely saturated. This strategy optimized the contrast and signal to noise for each filter over most of the brightness range and also compensated for varying filter

densities, so that white produced a uniform response.

There are well-known problems in using Vidicon television cameras for photometric measurements [2]. The dynamic brightness range of typical outdoor scenes, whether viewed live or on slides, generally exceeds the available 100:1 dynamic range of most television cameras. Quantitative comparisons of brightness observed through multiple color filters are frustrated by spectral variations in target sensitivity and by automatic gain circuitry built into the camera electronics. One way to avoid these problems is to digitize slides with a mechanical scanner using a photomultiplier. Time can then be directly traded for any desired combination of sensitivity, intensity resolution, signal/noise, dynamic range, and so forth. Lacking such hardware ourselves, we were fortunate to acquire over the ARPA net a small library of accurately digitized scenes from researchers at Carnegie Mellon University. These images were originally digitized on a Muir-head drum at the University of Southern California Image Processing Laboratory. The original data contained 8 bits/color at a spatial resolution of 600 X 820. These data were subsequently resampled down to 240 X 240 resolution. Some of these images were used in region growing experiments reported later.

2.1.2 Color Display

A RAMTEK* GX100 color video display was acquired and interfaced, replacing an Adage vector display used in earlier ISIS versions. The Ramtek system consists of an 18 inch color

TV monitor refreshed through D/A converters by a series of MOS shift registers. Each individually addressable shift register (or memory plane) stores 1 bit of brightness information for each discrete raster element. The system purchased by SRI contains a total of 14 memory planes which can be configured to produce two basic capabilities:

(1) Display a 256 X 256 element color image, each element displayed using 4 bits of intensity information each for red, green, and blue. Two overlays, bright red and bright green, are available for superimposing vector graphics, cursors, and so on without disturbing the underlying image.

(2) Display a 256 X 256 element grey scale image, each element displayed with 8 bits of brightness on a black and white monitor.

These capabilities represent an economic balance between the spatial and brightness resolution available from our sensors and the cost of additional memory.** Solid state memory was chosen so as to avoid difficulties in synchronizing multiple color planes because of skew and stability errors inherent in disk refreshed systems.

All of our work to date has been performed with the color monitor, although we anticipate several future applications that will benefit from the high resolution grey scale

compability. Qualitatively, the images are quite adequate for experimental work. Figure 1a illustrates the Ramtek's basic capabilities with ideal data. Figures 1b to 1d show displays from our principal experimental domains.

**The cost of each memory plane for 256 X 256 spatial resolution was approximately $1,000. Memory cost increases linearly with additional brightness resolution and as the square of spatial resolution. Thus, an additional bit of color resolution for each primary color adds $3,000 to the total cost, while doubling spatial resolution to 512 X 512 at 4 bit/color quadruples memory cost to $48,000.

---

*Ramtek Corporation, Sunnyvale, California 94086.

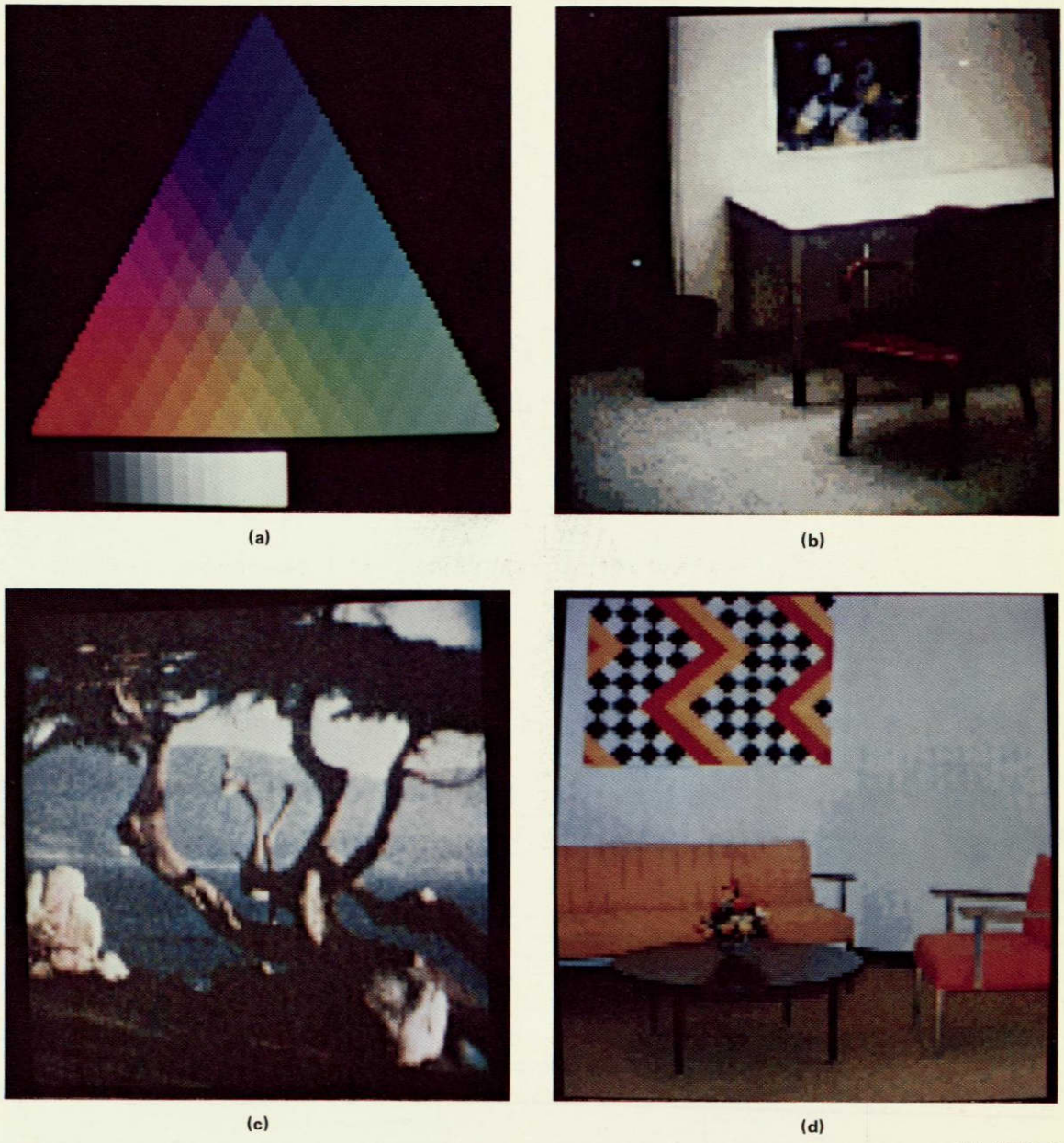


FIGURE 1 (a) SYNTHETIC COLOR TRIANGLE; (b) SRI OFFICE SCENE; (c) LANDSCAPE SCENE (MONTEREY, CALIFORNIA); (d) CARNEGIE-MELLON OFFICE SCENE

2.2 Software

The core of ISIS consists of the following basic software modules:

* Graphics support
* Primitive feature extraction operators
* Iconic and symbolic data structures
* General application programs (filtering, scanning, region growing, region classifying, statistics gathering, and so forth).

These components are being continually modified in an upward, compatible manner to improve efficiency and to increase the system's generality. Two improvements in the past year were particularly noticeable. First, the FORTRAN data structure used for storing region and sample descriptions has been replaced by a more compact and efficient structure implemented using special features of INTERLISP. Second, filtering and similar computationally expensive functions are now executed efficiently in FORTRAN. This has been made possible by software that allows arrays of image samples as well as conjunctive filter predicates to be communicated between LISP and FORTRAN.

A device-independent graphics package was implemented which allows displays with alphanumerics, points, vectors, grey

scale, and full color data to be developed in LISP, FORTRAN, or SAIL and displayed at run time on either the Ramtek, Adage, or Hewlett-Packard displays subject to hardware capabilities. Up to 100 display lists can be defined, which are manipulated independently.

Specialized support routines for the Ramtek allow arbitrary sized color image arrays to be shown at a specified scale and location on the display screen. Users can thus create montages of images at various stages of processing. Alphanumeric and vector overlays are automatically scaled to correspond with underlying image displays. Graphics can be output on any memory plane, although usually the red overlay plane is used. Informative color graphics such as bargraphs, histograms, false color image renditions, and so forth, can be synthesized conveniently with region coloring routines. Examples illustrating the use of these capabilities appear throughout this report. Detailed graphics software documentation is available from the authors.

3. PRIMITIVE OPERATORS

Scene analysis requires primitive operators that can distinguish among objects in a domain of interest. A combination of local attributes such as hue, height, and orientation provides adequate discrimination in simple room scenes, but not in landscapes. Natural colors tend to be weakly saturated with broad spectral peaks (see Figures 2a,b). Most of the boundary information usually associated with color is actually caused by brightness alone (Figure 2c). Height and orientation are more characteristic of plane surfaced room furniture than of irregularly contoured surfaces in natural scenes. (It is also much harder outdoors to obtain the range data needed for measuring these features.) Analyses of natural scenes consequently must use additional, global features derived primarily from visual data. The remainder of this section describes experimental primitives for three global features: texture, shape, and spatial context.

3.1 Texture

3.1.1 Introduction

Texture is an ill-defined property that gives nonhomogeneous regions a coherent appearance. Texture arises from regular (possibly statistical) variations in local attributes (such as brightness, hue, and saturation) and exists at all levels of description. Thus, a hillside of trees, a treetop, and the surface of a single leaf each have characteristic textures. One is usually

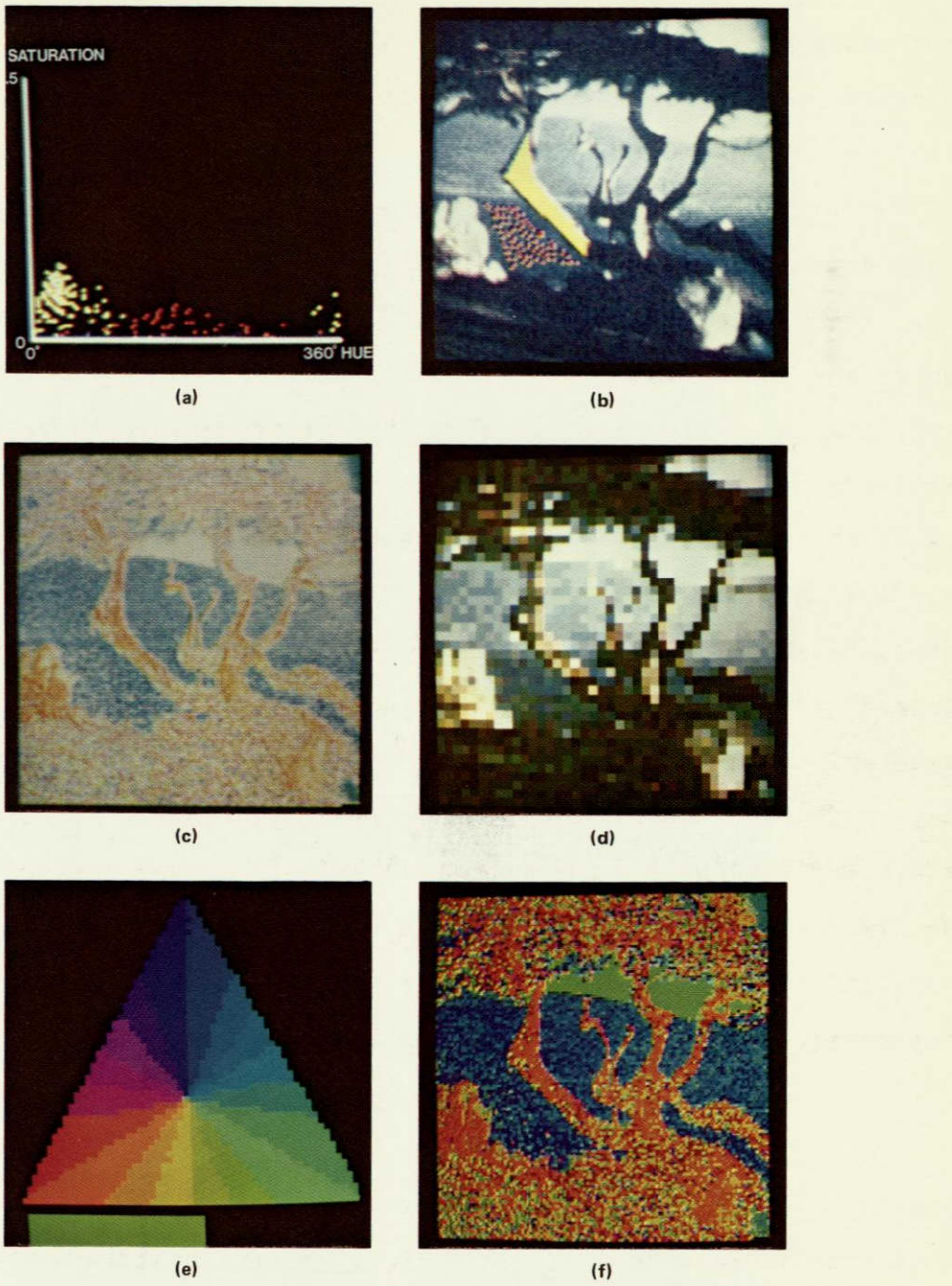


FIGURE 2 (a,b) COLOR CODED SCATTER PLOT OF HUE VERSUS SATURATION FOR REGIONS DESIGNATED IN FIGURE 2b; (c) LANDSCAPE SCENE FROM FIGURE 1c WITH BRIGHTNESS INFORMATION REMOVED (BRIGHTNESS NORMALIZATION ACHIEVED BY DIVIDING THE RED, GREEN, AND BLUE COLOR COMPONENTS OF EACH PICTURE ELEMENT BY THE SUM OF THOSE COMPONENTS); (d) LANDSCAPE SCENE SAMPLED TO 40 x 40 RESOLUTION; (e) 20 DEGREE QUANTIZATION INTERVALS DISPLAYED ON COLOR TRIANGLE; (f) LANDSCAPE SCENE WITH EACH SAMPLE DISPLAYED IN THE COLOR OF THE 20 DEGREE INTERVAL TO WHICH IT IS CLASSIFIED

most aware of the gross texture of the smallest scene elements involved in an articulated description.

The texture detail available in a digitized image depends on viewing scale (i.e., magnification) and spatial sampling density. The level of texture needed to distinguish, say, a region of grass from a region of water has not been clearly determined. It would, of course, be preferable to rely on macro texture where grass might be characterized as a region containing green, brown, and yellow patches. One could then sample very coarsely (e.g., 40 X 40 or less) and avoid the mass of data needed to characterize the individual blades. We thus concur with Hanson and Riseman [3] who questioned Bajcsy's arguments for using a quantization grid several times finer than the finest texture element [4]. Figure 2d shows the same scene as Figure 1c (120 X 120 resolution), sampled at 40 X 40 resolution. Viewed at a distance of, say, 15 feet, this figure is quite recognizable. Therefore, qualitatively it probably contains sufficient texture data to characterize its component objects.

Texture, unlike brightness and hue, is not a monolithic attribute and cannot generally be expressed by any single functional representation. Investigators have used a wide variety of features to classify or distinguish particular textures, but have been unable to formalize how features should be selected for a particular class of scenes. ISIS is a useful tool for determining appropriate texture features experimentally; operators can be applied to selected regions of a scene individually and in weighted feature

vectors to test discrimination.

3.1.2 Texture Operators

We are accumulating a library of commonly used texture operators and procedures (Table 3.1). These operators are grouped into two classes: micro-textures and macro-textures. Micro-textures are characterized by statistical distributions of brightness, hue, and saturation at the picture element level. Macro-textures are composed of groupings of elementary regions (i.e., regions of homogeneous color and brightness) and are characterized by distributions of shape, density, spatial arrangement, and micro-texture of those regions.

Table 3.1

A BASIC LIBRARY OF TEXTURE OPERATORS

Micro Textures

1. Distribution statistics for brightness, hue, and saturation (taken directionally or nondirectionally):

   a. Mean
   b. Mode
   c. Standard deviation
   d. Skew
   e. Kurtosis

2. Features derived from directional spatial dependency arrays [5]:

   a. Energy
   b. Entropy

*3. Edge density and directional edge density (vertical edge density/horizontal edge density) [6]:

*4. Statistics derived from Fourier power and phase spectrum [4]:

   a. Peak power, bimode, and so on.
   b. Linguistic features: monodirectional, linear, bidirectional, blobs, and so on.

*Macro Textures

Statistics derived from distributions of elementary region properties [7]:

   a. Shape (based on moments)
   b. Size (perimeter, area)
   c. Density (average number of regions/unit area)

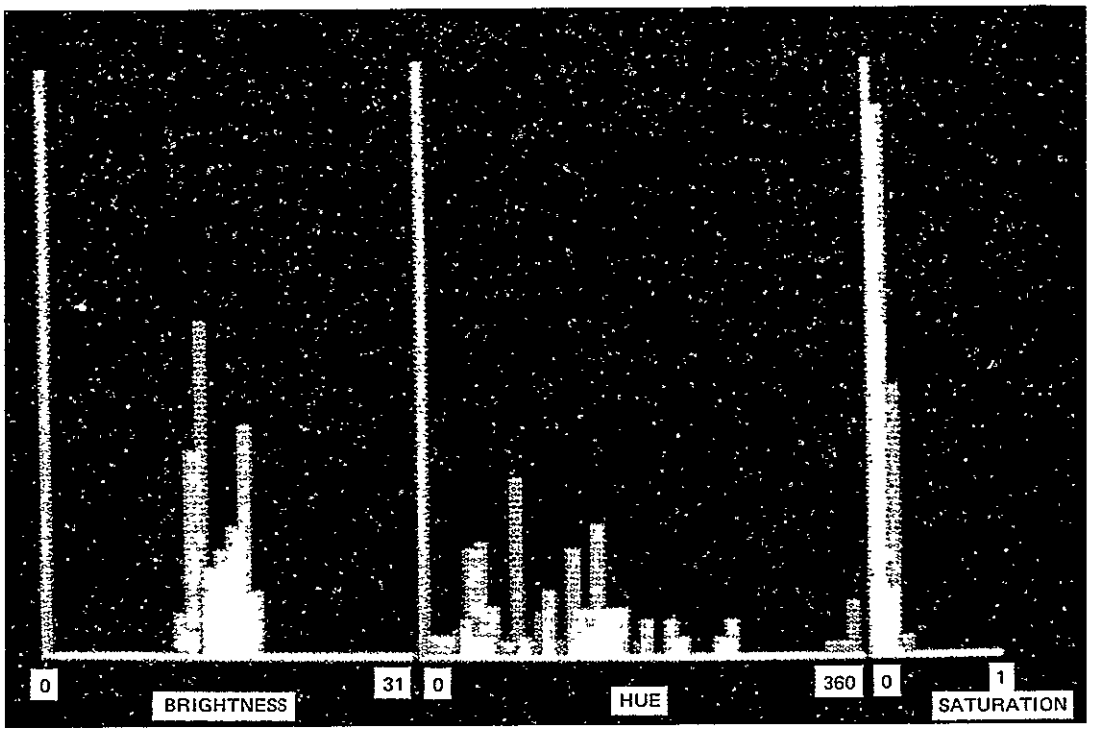
*These operators have not yet been implemented on our system.

3.1.3 Texture Displays

A number of special displays have been formulated to help visualize the effectiveness of various texture operators. An experimenter can circle two regions of the scene and obtain superimposed histograms of hue, saturation, and brightness (see Figure 3). He can also obtain superimposed two-dimensional scatter plots of hue versus saturation for multiple regions. (The yellow and red scatter points in Figure 2b correspond respectively to the yellow and red regions designated in Figure 2a.) He can display the original image sampled to an arbitrary resolution (Figure 2d). Furthermore, each sample can be shown with the average or modal brightness or hue of the surrounding area (Figures 4, 5). Spatial dependency arrays (giving the probability of encountering a transition between two specified brightness levels when scanning the image horizontally, vertically, or diagonally) can be displayed for selected regions as two-dimensional brightness arrays rather than as large arrays of meaningless numbers (Figure 6).

3.1.4 Discussion

A systematic evaluation of texture operators is not one of our objectives, although a facility such as ISIS would be helpful in such an undertaking. From a cursory examination, it appears that statistics for spatial texture implemented so far do not discriminate substantially better than a comparison of brightness, hue, or saturation distributions using standard tests of significance such as chi-square or Kolmogorov-Smirnov (KS). Encouraging



(a)

(b)

TA-8721-4

FIGURE 3 (a,b) OVERLAPPING HISTOGRAMS OF BRIGHTNESS, HUE, AND SATURATION FOR REGIONS OUTLINED IN FIGURE 3B

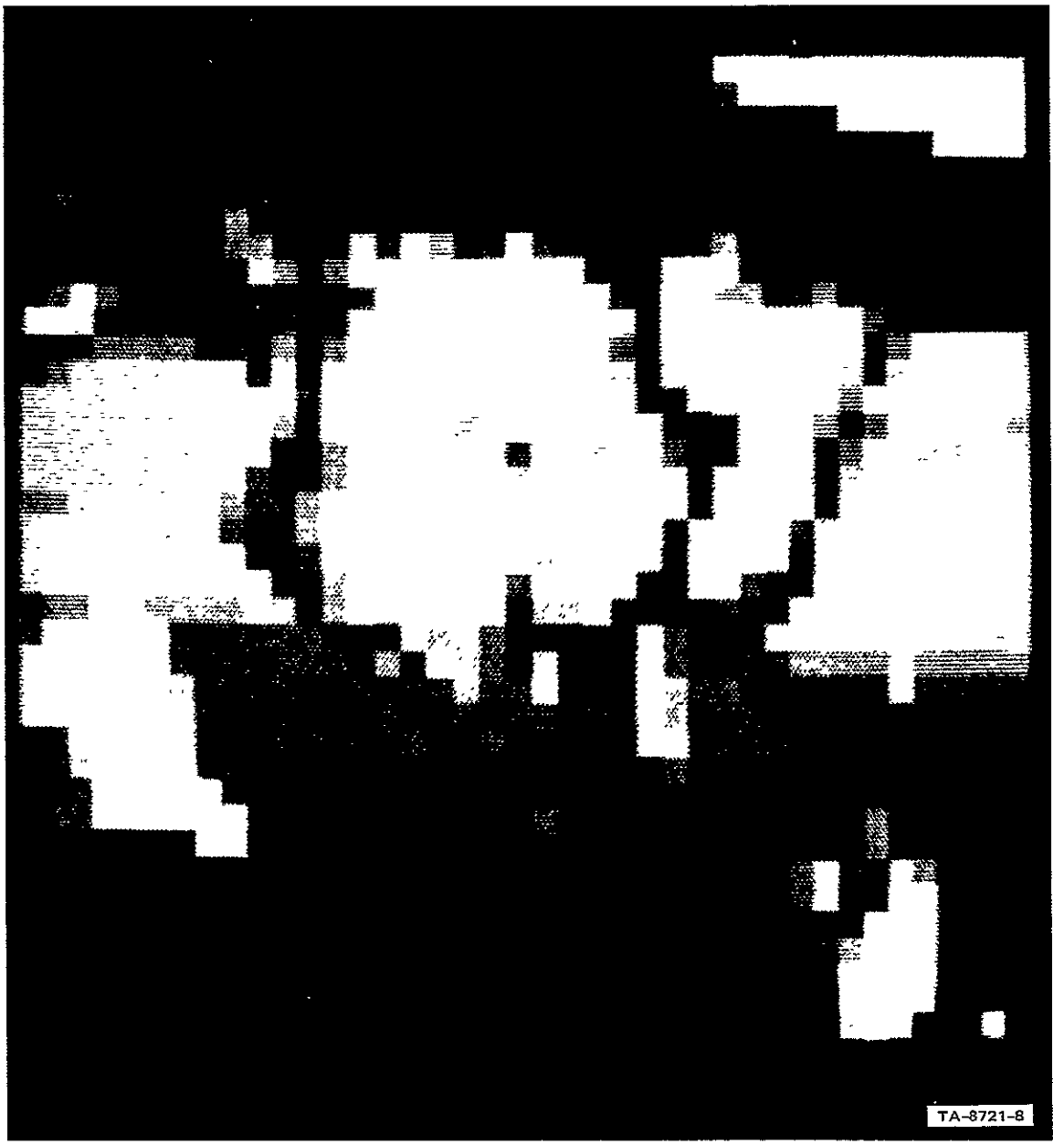


FIGURE 4 SAMPLED LANDSCAPE SCENE WITH EACH SAMPLE DISPLAYED AT THE AVERAGE BRIGHTNESS OVER A 3 x 3 NEIGHBORHOOD

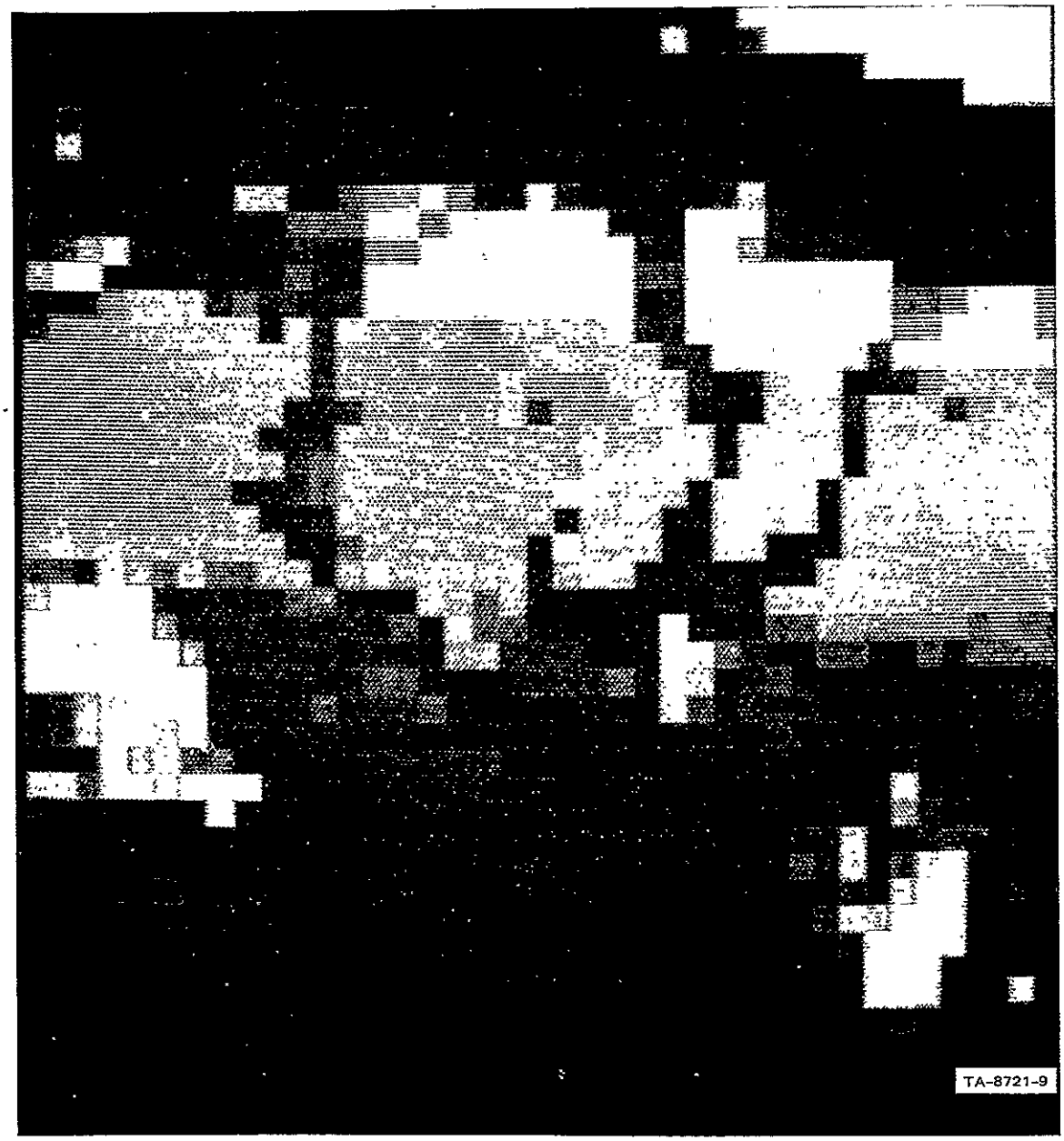


FIGURE 5 SAMPLED LANDSCAPE SCENE WITH EACH SAMPLE DISPLAYED AT THE MODAL BRIGHTNESS OVER A 3 x 3 NEIGHBORHOOD

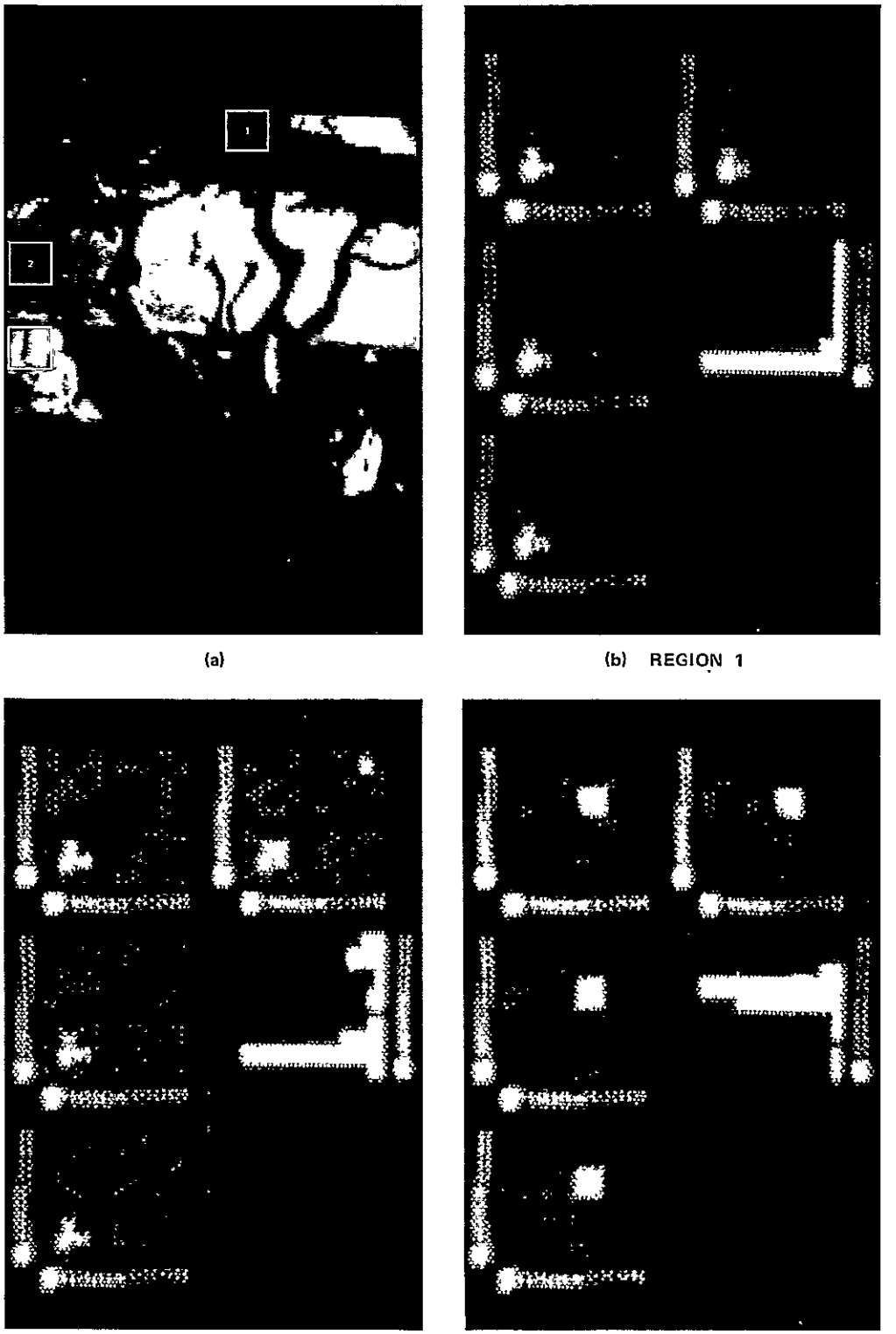


FIGURE 6 SPATIAL DEPENDENCY ARRAYS FOR REGIONS DESIGNATED IN FIGURE 6a

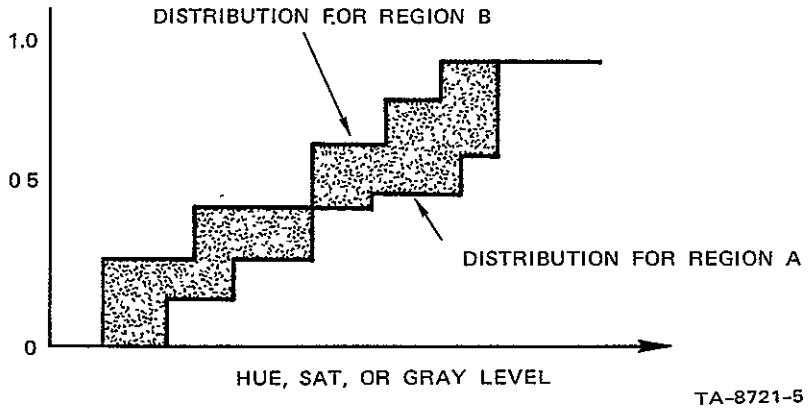


FIGURE 7 MODIFIED KOLMOGOROV—SMIRNOV SIMILARITY CRITERIA: TWO CUMULATIVE DISTRIBUTIONS ARE COMPARED ON THE BASIS OF AVERAGE MAGNITUDE DIFFERENCE, SHOWN SHADED (After Muerle and Allen [8]).

experimental results have been obtained using a modification of the KS criteria first proposed by Muerle and Allen [8] and illustrated in Figure 7.

A promising future direction is the use of ISIS to deduce ad hoc characteristics that distinguish particular textures in a limited scene domain. For example, a region might be adequately characterized, at the micro-texture level, simply as the set of samples with a prescribed proximity to samples having a distinguished hue (the detailed hue distribution of the selected points being unimportant). In one particular scene, regions of sky and lake both contained many samples with virtually identical blue hues. However, in the lake, the blue samples were liberally interspersed with distinctive green samples. At the macro-texture level, a region could be described in terms of distinguishing attributes of component regions, as when describing grass as a region containing green, yellow, and brown blobs. A particularly simple macro-texture descriptor is the number or density of smaller regions contained in a standardized window. In Figure 8, for instance, the river and bushes are partitioned into many small regions, while grass, sky, and trees are represented by a few large regions.

Although texture should someday significantly enhance color discrimination, it will not always allow unique interpretation. Regions of ground, dark tree bark, and treetop in Figure 1c are virtually indistinguishable to the human eye when viewed through small slits in a mask. The same confusion applies to regions of mountain and sea and of rock and light tree bark. Clearly, shape and spatial context are also necessary.

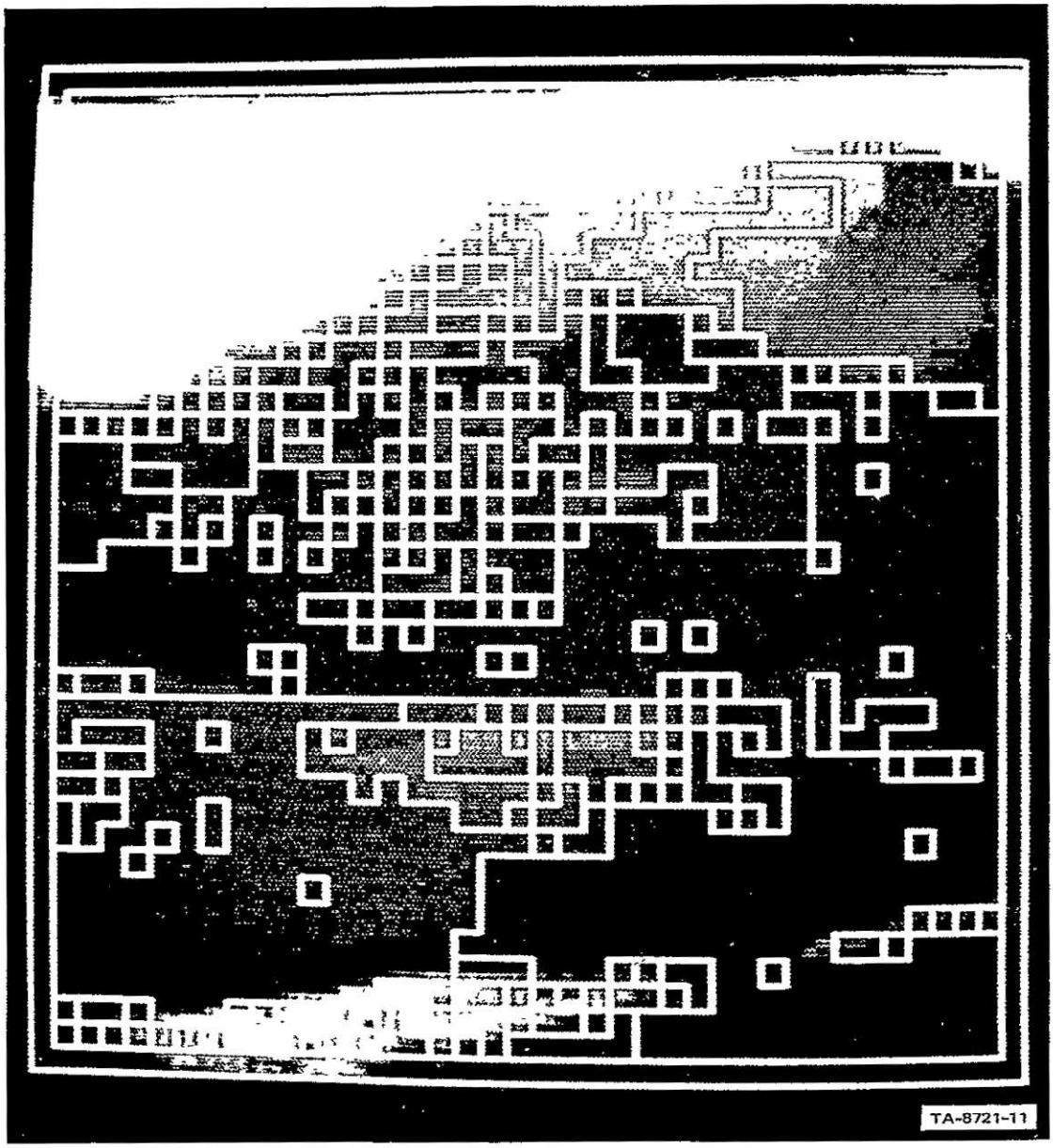


FIGURE 8 RIVER SCENE SEGMENTED INTO 350 RELATIVELY HOMOGENEOUS REGIONS, ILLUSTRATING THE USE OF REGION DENSITY AS A TEXTURE FEATURE

3.2 Shape

Shape, like texture, is not a monolithic attribute and therefore is not amenable to universal representation. Table 3.2 lists some two dimensional shape primitives. So far, only the first two primitives have been implemented. The ratio of (vertical perimeter/horizontal perimeter) was tested and proved effective in distinguishing thin, vertically elongated regions of "treebark" from non-elongated regions of "ground" and "treetop" in partitioned landscape scenes. (Sections of jagged boundary with horizontal or vertical extent less than two were ignored in computing this ratio.)

Table 3.2

SHAPE PRIMITIVES FOR REGIONS

A) Global Shape

1. Compactness: (perimeter squared/area)

2. Orientation: (Vertical perimeter/horizontal perimeter)

3. Aspect Ratio: (length/width of tightest bounding rectangle)

4. Moments

5. Skeleton characteristics (e.g., branching structure) [9]

B) Boundary Descriptors (location and global)

a. Curvature Points [10]

b. Fourier expansion of boundary curve [11, 12]

c. Chain code features [13]

d. Linguistic descriptions [14, 15]

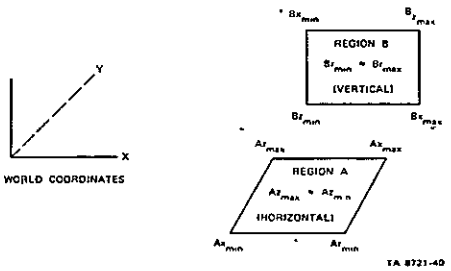
3.3 Spatial Relations

Spatial context is an important factor in resolving interpretation ambiguities. Procedural representations have been implemented for some common three dimensional spatial relationships between two regions, based on the relative world coordinates of vertices in their polygonal boundaries. These representations are described in Table 3.3a and demonstrated in Table 3.3b using the test regions in Figure 9.

The above representations were originally developed for room scenes, assuming availability of range data. The accuracy of our developmental rangefinder has, thus far, been disappointing: 6 inches/10 feet. However, range accuracy is much less critical in determining global relations than in determining local attributes, such as surface orientation. Moreover, all of the relations except for planarity can be reformulated in terms of two-dimensional image coordinates for standard eye level views.

Table 3.3A

REPRESENTATIONS FOR SPATIAL RELATIONS BETWEEN TWO PLANAR SURFACES
GIVEN 2 REGIONS--A (e.g., a Horizontal Chair Seat) and B (e.g., a Vertical Chair Back)



1 Left of/Right of

Let Axmin, Axmax = minimum and maximum X image coordinates of boundary points of Region A
Bxmin, Bxmax = minimum and maximum X image coordinates of boundary points of Region B
then

A Region A is left of Region B

iff Bxmin + Bxmax > Axmin + Axmax
and Bxmin > Axmax or

$$\frac{\text{Max (Axmin, Bxmin)} - \text{Axmax} - 1}{\text{Min (Axmax - Axmin, Bxmax - Bxmin)} + 1} \leq .49$$

(The last condition provides a reasonable interpretation of the concept "left" in cases where Regions A and B partially overlap)

B. Region A is right of Region B

iff Bxmin + Bxmax ≥ Axmin + Axmax
and Axmin > Bxmax or

$$\frac{\text{Min (Axmax, Bxmax)} - \text{Axmin} + 1}{\text{Min (Axmax - Axmin, Bxmax - Bxmin)} + 1} \geq .49$$

2 Below/Above

Let Azmin = Azmax = height at maximum and minimum Y image coordinates of boundary points of Region A (horizontal surface)
Bzmin, Bzmax = height at maximum and minimum Y image coordinates of boundary points of Region B
then

A. Region A is below Region B

iff Bzmin + Bzmax > Azmin + Azmax
and Bzmin > Azmax or

$$\frac{\text{Max (Azmin, Bzmin)} - \text{Azmax} - 1}{\text{Min (Azmax - Azmin, Bzmax - Bzmin)} + 1} \geq .1$$

B. Region A is above Region B

iff Bzmin + Bzmax ≥ Azmin + Azmax
and Azmin > Bzmax or

$$\frac{\text{Min (Azmax, Bzmax)} - \text{Azmin} + 1}{\text{Min (Azmax - Azmin, Bzmax - Bzmin)} + 1} \geq .1$$

(An additional relation, directly above/directly below may be defined by requiring that the regions involved not be to the right, left, in front, or in back of each other.)

3. Front/Back

Let Armin, Armax = maximum and minimum range of boundary points of Region A
Brmin, Brmax = maximum and minimum range of boundary points of Region B (vertical surface)
then

A. Region A is in front of Region B

iff Brmin + Brmax > Armin + Armax
and Brmin > Armax or

$$\frac{\text{Max (Armin, Brmin)} - \text{Armax} - 1}{\text{Min (Armax - Armin, Brmax - Brmin)} + 1} \geq .9$$

3 Front/Back (Concluded)

B. Region A is in back of Region B

iff Brmin + Brmax ≤ Armin + Armax
and Armin > Brmax or

$$\frac{\text{Min (Armax, Brmax)} - \text{Armin} + 1}{\text{Min (Armax - Armin, Brmax - Brmin)} + 1} \leq .9$$

4. Coplanar

Let PLA, = least square planar surface fit to boundary points of Region A
PLB, = least square planar surface fit to boundary points of Region B
then

Region A and Region B are coplanar iff the following criteria hold:

(1) The surface normals of PLA and PLB must be parallel to within 10 degrees.

(2) Each local plane must intercept the same coordinate axis (X, Y, or Z) closest to the origin.

(3) These (most reliable) intercepts must agree within 10%

(These above criteria compensate heuristically for uncertainties in range data.)

Table 3.3b

RELATIONS OF SURFACES IN FIGURE 9 USING REPRESENTATIONS IN TABLE 3.3a
(Table lists relations of object 1 to object 2)

where A = above F = in front L = left
BL = below BK = in back R = right

	OBJECT 2						
OBJECT 1	Chairback	Chairseat	Door	Picture	Tabletop	Wall	Wastebasket
Chairback	---	A,BK	R	R,BL	F	R,BL	R,A
Chairseat	R,F	---	R	R,BL,F	BL,F	R,BL	R,A
Door	L	L	---	L	L	L	L,BK
Picture	L,A	L,A,BK	R	---	A,BK	R	R,A
Tabletop	BK	A,BK	R	BL,F	---	R,BL	R,A
Wall	L,A	L,A	R	L	L,A	---	R,A,BK
Wastebasket	L,BL	L,BL	R,F	L,BL	L,BL	L,BL,F	---

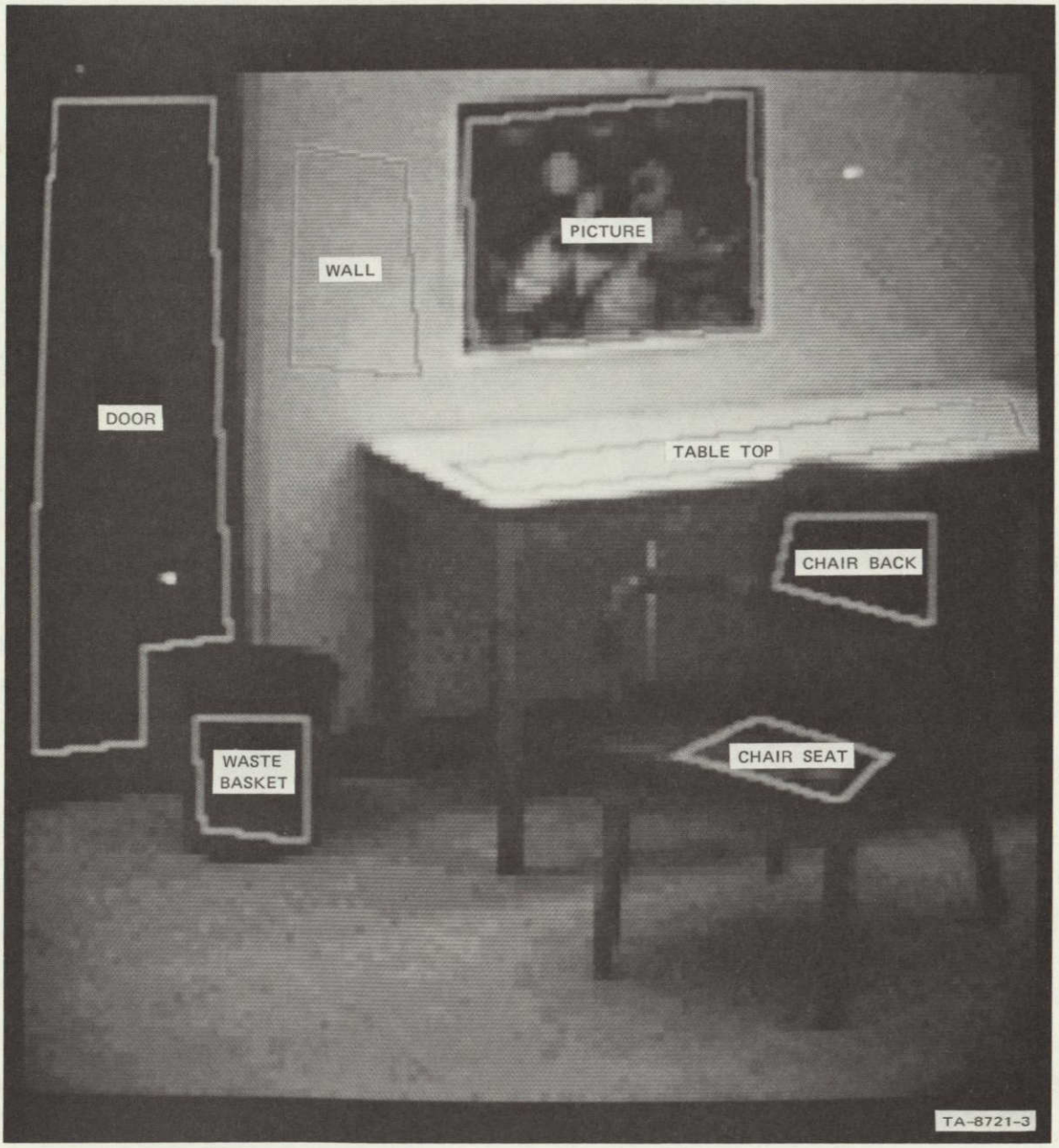


FIGURE 9 REGIONS USED TO TEST SPATIAL RELATIONS (TABLE 3-b)

4. A REGION ANALYSIS SUBSYSTEM FOR ISIS

4.1 Introduction

Most analyses of natural scenes first attempt to partition the image into coherent regions corresponding to known objects [4, 16, 17, 18]. Regions provide a convenient basis for semantic analysis by reducing both the amount of detail and the ambiguities of interpretation found at the picture element level.

Several models of region analysis have appeared in the literature. Bottom-up methods begin by exhaustively partitioning a scene into elementary atomic regions and proceed by merging together adjacent regions with weak common boundaries [17, 19]. Top-down methods begin with a single region encompassing the entire image and successively segment it into smaller, homogeneous subregions [20]. These methods can be combined with increased efficiency by allowing both merging and segmentation to proceed from an initial intermediate level partition obtained by imposing an arbitrary grid [21, 22]. Regions can also be grown independently from starting kernels selected with semantic information [23].

This section presents the region analysis capabilities being developd for ISIS. We first evaluate some nonsemantic region analysis techniques we and other researchers have developed and then discuss further improvements based on semantics (problem-dependent knowledge) and user interaction.

4.2 Basic Model

Region analysis procedures were developed for ISIS using a bottom-up paradigm evolved from Brice/Fennema. There are two principal stages of processing: first partition and region growing. The purpose of first partition is to obtain an initial segmentation of the image which is conservative in the sense that each region consists of picture elements belonging to only one object. In the region growing stage, adjacent regions with similar characteristics are merged into a single region to further simplify the organization. The desired result is a set of regions corresponding to distinct objects in the scene.

The most conservative first partition is obtained by making every picture element a separate region. However, region analysis is computationally expensive, making it infeasible to process this amount of detail in a reasonable amount of time. The common practice is to first sample the picture to reduce resolution and then immediately merge adjacent samples with identical characteristics. Brice/Fenema sampled a 120 X 120 image with 16 levels of brightness down to 60 X 60 resolution, and then combined adjacent elements with equal brightness, yielding about 1,000 elementary regions in a blocks world scene. When data are finely quantized or multidimensional (i.e., color and range data are available), demanding strict equality of all attributes can lead to an unnecessarily large number of regions, many caused by quantization noise. In such cases it may help to first classify each sample into a small number of categories and then treat picture elements assigned

to the same category as identical. Yakimovsky, for example, sorted each image sample into one of 25 color categories covering distinctive combinations of hue and saturation in his outdoor scenes [17]. Lieberman obtained an adequate partition of two simple landscapes using only seven categories of hue [4]. Experiments have been conducted with several sampling and quantization schemes for obtaining first partitions of landscape and office scenes, and a few methods have been proved empirically to be adequate. Our findings are reported in Section 4.4.

Region growing proceeds by serially selecting the pair of adjacent regions in the current organization which are globally "most alike," and merging these into a single region. The order in which regions are merged is determined by a function that compares the similarity of a given pair of adjacent regions with the similarities of other pairs of regions that remain as candidates to be merged. A variety of criteria for region similarity have been used, including average brightness contrast [11, 19] and average color contrast [17]. Brice/Fennema also used a weighting factor that weakened the effective contrast of regions that meet along meandering boundaries, a common characteristic of quantization contours [19]. In our system the function that determines similarity (and hence merge priority) of adjacent regions is a separate program module allowing experimentation with a variety of criteria that may include problem dependent considerations. Experimental results with various similarity functions are presented in Section 4.5.

Various algorithms have been used to control the

order and extent of merging. Brice/Fennema merged boundaries in arbitrary order so long as the weighted strength was below an experimentally determined absolute threshold. The resulting partition was dependent on the order of merging. This prompted Yakimovsky to merge boundaries on a weakest-first basis, terminating when either an absolute difference threshold or a minimum number of regions was exceeded. Another method, proposed by Freuder [24], was to merge two regions, without regard for absolute boundary strength, whenever both were more similar to each other than to any other neighbors. The region growing procedures developed for ISIS follow Yakimovsky's merge strategy by keeping pairs of adjacent regions on a priority queue, and always first merging across the globally weakest boundary.

The basic region analysis process is illustrated by an example depicted in Figures 10 through 13. The first partition stage is based on the brightness values associated with a 40 X 40 sampling of the image. Region pairs are added to the priority queue, using a similarity measure based on average absolute brightness and color difference across their common boundaries. Region pairs are then removed from the queue one at a time, in order of similarity, and a merge performed in the region data structure. The first partition (Figure 10) yielded 806 initial regions (half as many regions as individual picture elements). The results of subsequent merging are illustrated with 600, 450, and 250 regions remaining.

All region-based techniques that effectively analyze

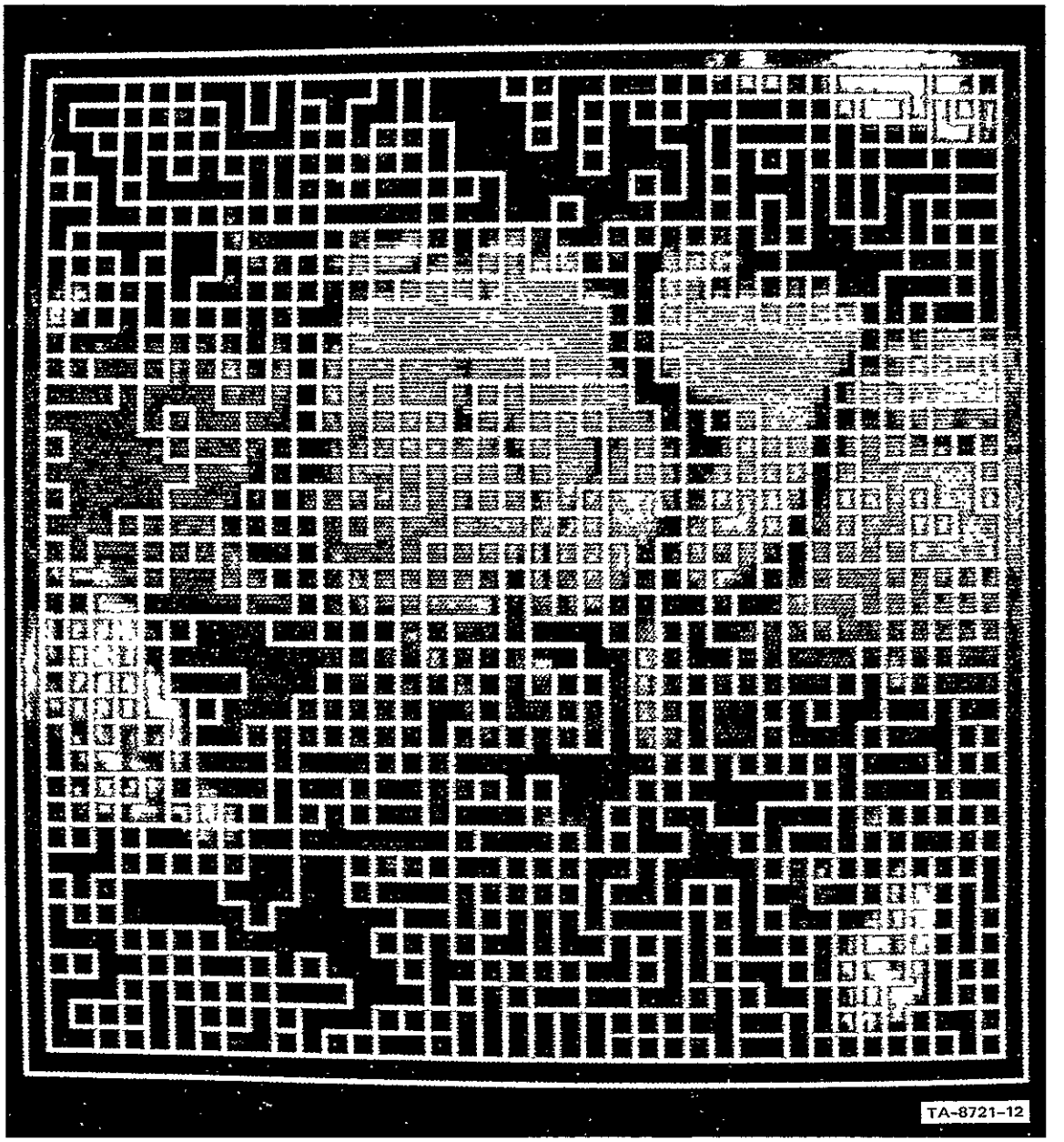


FIGURE 10 FIRST PARTITION OF LANDSCAPE SCENE (806 REGIONS)

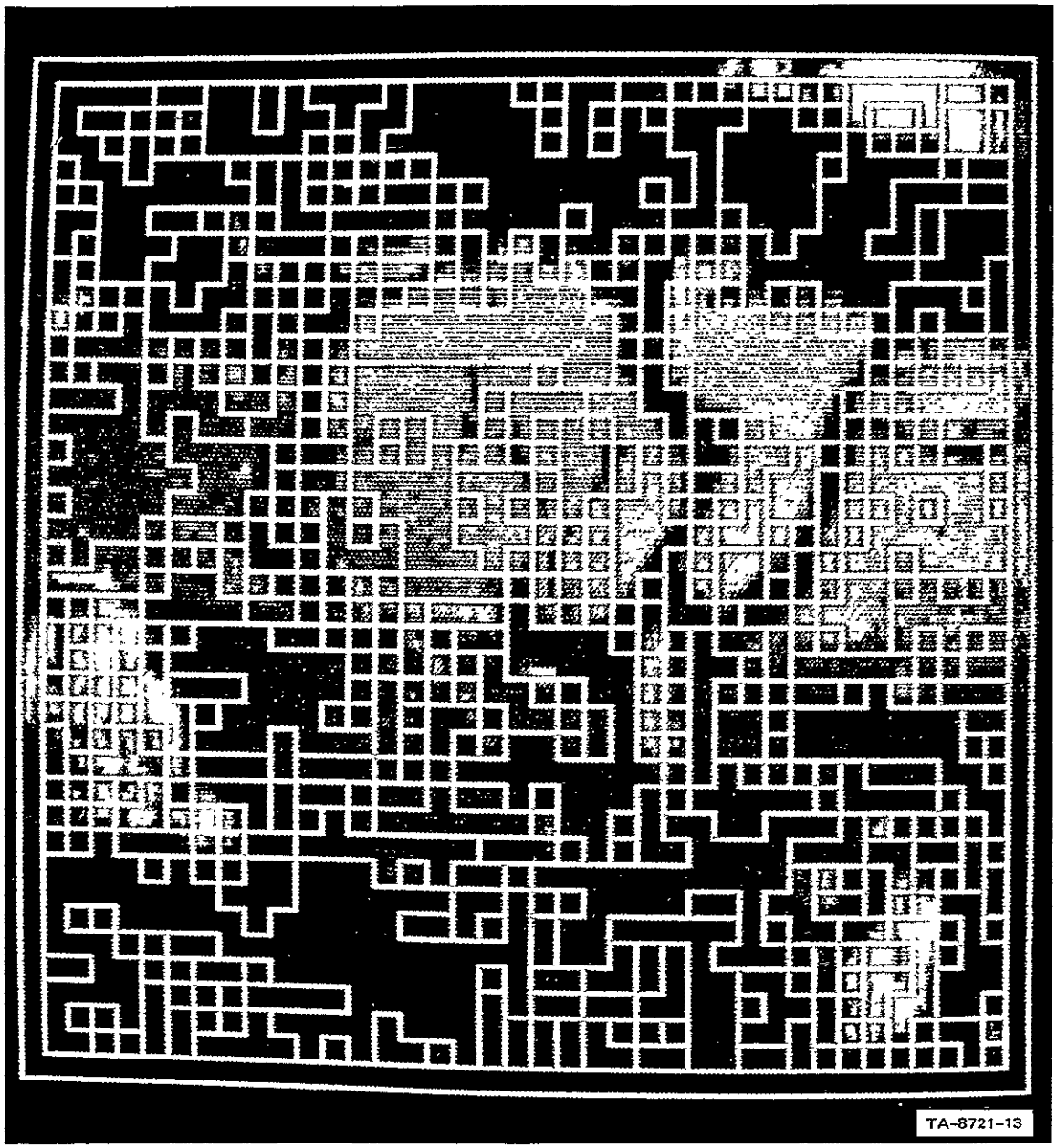


FIGURE 11 PARTITIONED LANDSCAPE SCENE AFTER 206 MERGES (600 REGIONS REMAINING)

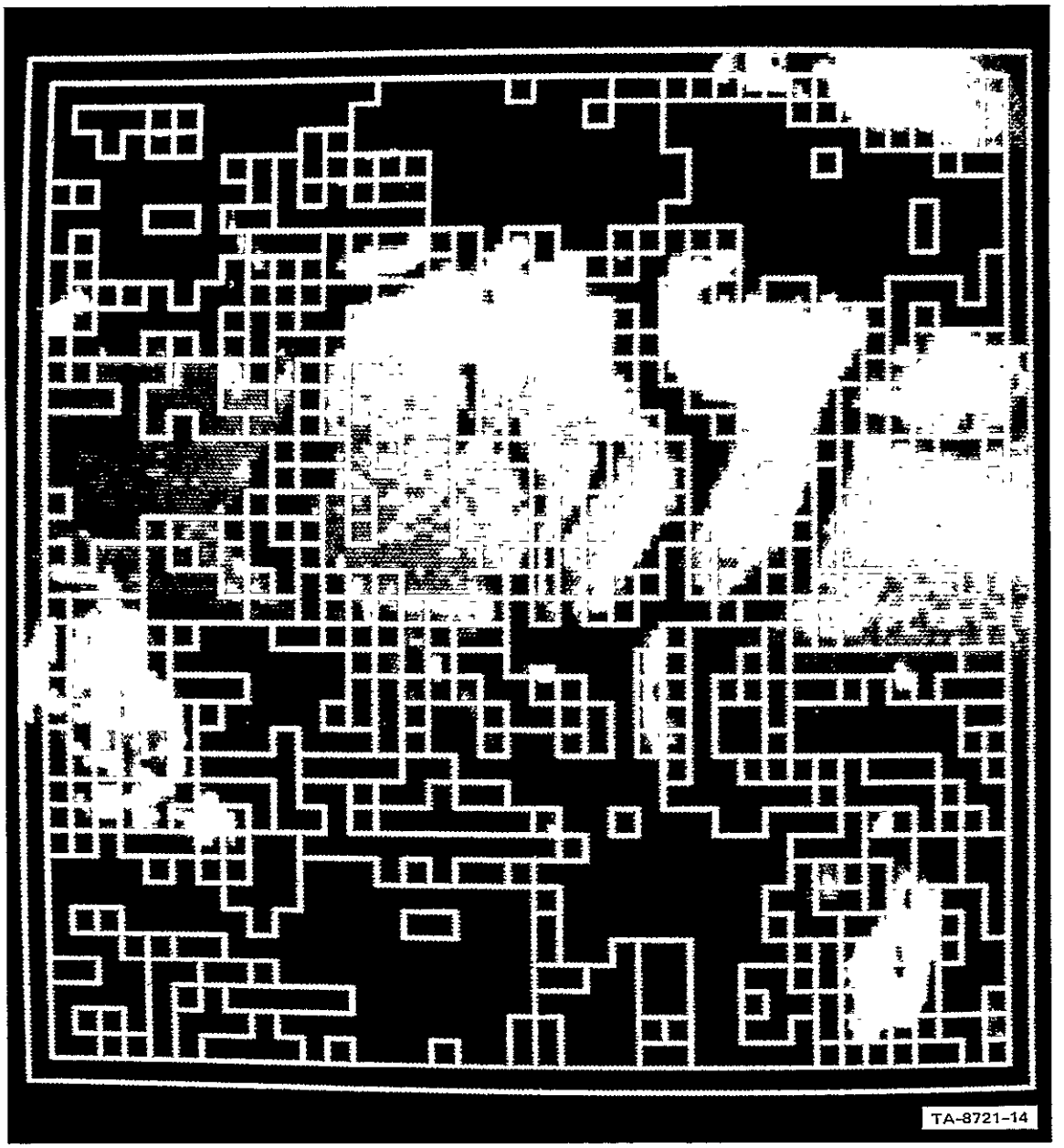


FIGURE 12 PARTITIONED LANDSCAPE SCENE AFTER 150 ADDITIONAL MERGES (450 REGIONS REMAINING)

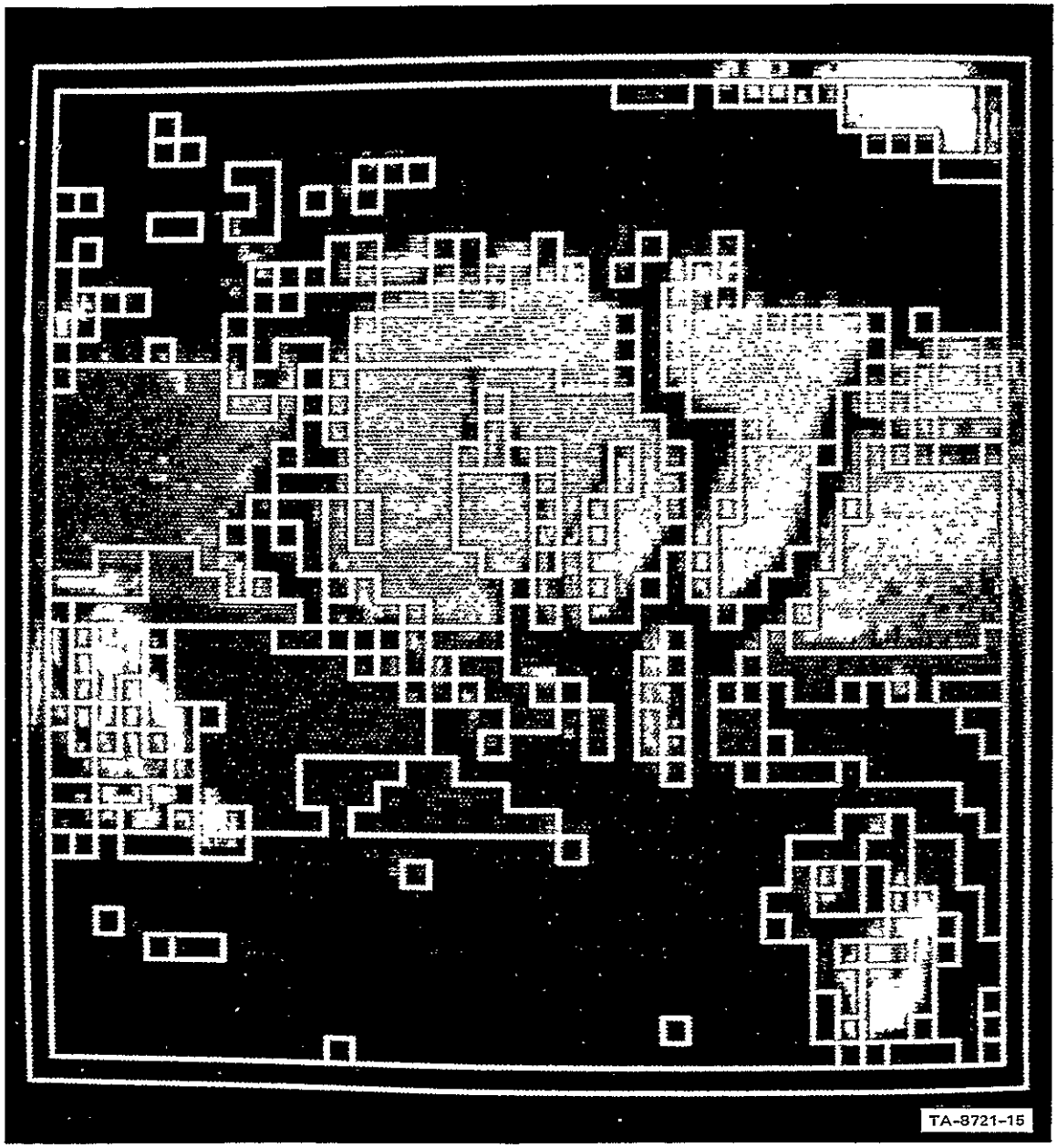


FIGURE 13 PARTITIONED LANDSCAPE SCENE AFTER 200 ADDITIONAL MERGES (250 REGIONS REMAINING)

nontrivial scenes make use of context-sensitive rules based on problem-related knowledge. We will refer to such rules under the general heading of semantics.

Semantics have been introduced into merging decisions in a variety of ways. The Brice/Fennema weakness heuristic contains an implicit semantic statement: edges of objects are straight in the blocks world. After an initial phase of merging based on nonsemantic similarity criteria, Yakimovsky's program continues merging with a semantic measure of boundary strength used to set merge priorities. This measure used a Bayesian model to compute the likelihood that two adjacent regions had the same interpretations, based on both their individual attributes (e.g., color, size, and texture) as well as on attributes of their common boundary (e.g., length, shape, orientation).

Lieberman's program, after grouping samples into regions of homogeneous hue, identified the largest regions and then invoked semantic (object dependent) procedures to extend and refine boundaries. For example, regions thought to be sky were extended to adjoining regions having a homogeneous texture, while regions thought to be ground were extended to adjacent regions having a vertically receding texture gradient. The Harlow and Eisenbeis program used semantic knowledge (primarily expected brightness and image location of each region) to select starting locations for regions corresponding to the principal anatomic features in a chest x-ray [23]. Initial regions were formed by independently merging starting kernels with adjacent samples of similar brightness. The

similarity threshold differed for each feature depending on expected brightness variations. This stage terminated when the initial regions had reached a significant size, but before they grew together, subsequent merging was constrained by a set of semantic "structure rules" that tried to ensure that the final regions complied with known spatial relationships among the features.

Semantic considerations are by definition related to specific problems or picture domains, and hence they are inherently ad hoc. For this reason, man-machine interaction is the appropriate mechanism for interjecting semantic information that guides the region growing process and for experimenting with different semantic rules and determining empirically their usefulness in a specific problem domain. With an interactive facility, the researcher can propose a set of semantic rules and instruct the system to continue processing until a specified state of the world or error condition is reached. At this point, he can investigate the properties of regions and their relationships and modify the knowledge base guiding the analysis. It is here that we feel the region analysis techniques developed for ISIS constitute a significant original contribution to the field.

4.3 Interactive Features

Some interactive control features are provided to aid in the formulation of effective region growing strategies, and in the selection of appropriate stopping criteria. Merges can be performed in "step" mode, one merge at a time, allowing the dynamic

characteristics of a priority function to be observed. Proposed merges are indicated on the display prior to execution. The user may then simulate alternative merging criteria by modifying the queue manually. For example, regions can be selectively deleted from the queue, thus becoming unmergeable. The user can study the dynamics of a particular erroneous merge by requesting the system to enter step mode whenever a merge is proposed within a designated rectangular window.

The ability to build and modify the priority queue interactively is also useful for experiments in cooperative (man-machine) region analysis. For instance, the queue may be originally built with a few manually selected regions and their "neighbors" (adjacent regions). These regions then serve as "kernels" from which all subsequent new regions grow.

4.4 First Partition Experiments

The objective of first partition is to group together adjacent picture samples with similar attributes, so as to obtain the fewest initial regions without risking a false merge. In the introductory example, the picture was first sampled to 40 X 40 resolution; then adjacent samples with identical brightness were combined to form homogeneous atomic regions (Figure 10). This section reports on attempts to obtain improved first partitions using different sampling functions and different criteria for judging the similarity of adjacent samples.

4.4.1 Experiments on Sampling

Processing time considerations dictate that region analysis be performed at the lowest resolution that preserves important details in the image. The experiments reported in this section compare first partitions obtained with straight sampling, modal sampling, and mean sampling. All experiments were performed on gray scale images using a 40 X 40 rectangular sampling grid. (In scenes with periodic texture, a random sampling strategy is recommended to avoid aliasing effects.) In modal sampling (Figure 14), the gray level of each grid point is taken to be the most frequently occurring value of gray level for nearby points. The number of initial regions obtained in this manner is significantly reduced (by about one-third), because a lot of small "noise" points in the treetop and ground disappear. However, the fine detail in the small central branch of the tree is lost, and the separation between

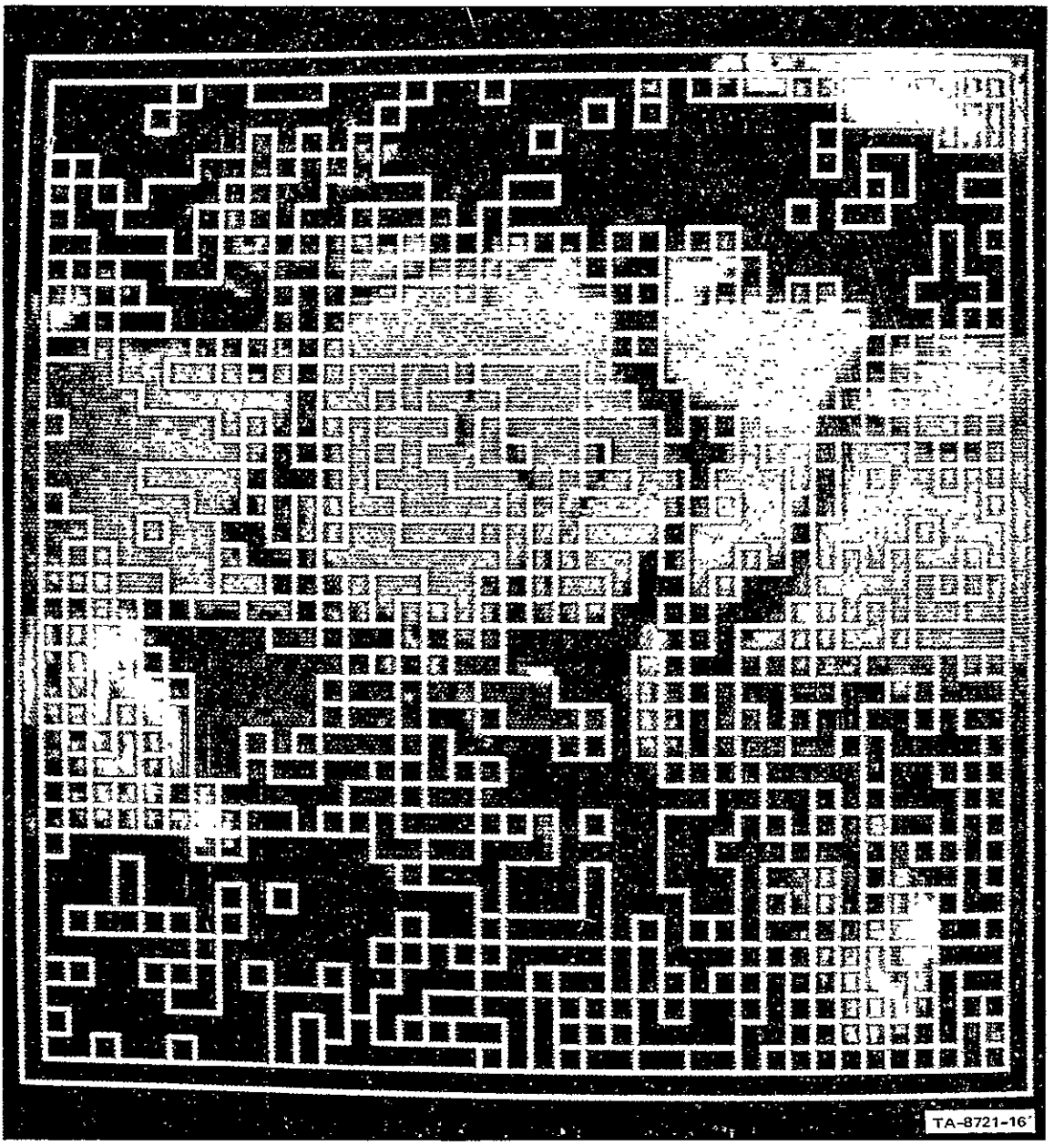


FIGURE 14 FIRST PARTITION OF LANDSCAPE SCENE PRODUCED FROM MODAL SAMPLES OF FIGURE 5 (536 REGIONS)

the ground and the treetrunk are lost by the modal smoothing. Sampling the mean gray-level in a small neighborhood around each grid point is a poor technique because of its extreme tendency to smooth discontinuities, as seen in Figure 4.

We concluded that first partitions based on a simple sampling of the gray scale image taken through a neutral density filter contain all of the information needed to conservatively partition most real indoor and landscape scenes. This conclusion is supported by the absence of most boundaries in images which have had brightness information removed by normalization (Figure 2c).

4.4.2 Experiments on Classification

The experiments presented above relied solely on brightness information for subdividing the image. The resulting partitions seem overconservative, with many objects unnecessarily fragmented. Since the region growing stage is quite time consuming, we were motivated to see whether additional sensory modalities (i.e., color and range data) could be used in conjunction with brightness to reduce the number of initial regions.

4.4.2a Nonsemantic Classification

In the following experiments, the hue of each picture sample was quantized into 20 degree intervals as shown in Figure 2e. (The hue of each sample is expressed as a

spectral angle from 0 to 360 degrees based on a transformation of the raw color data given in Appendix B of Reference 1.) Regions were then formed by grouping adjacent samples with hues falling in the same interval. Figure 2f is a false color presentation of Figure 1e, with each sample displayed in the hue of the 20 degree interval to which it is classified. Figure 15 shows the partition resulting from this classification.

Twenty degree hue intervals were thought to be conservative compared with the far cruder quantizations employed by both Yakimovsky and Lieberman. The resulting partitions, however, were consistently worse than brightness partitions. Major leaks occurred between semantically distinct regions in both indoor and outdoor scenes. Moreover, in outdoor scenes, because of textural irregularities, the total number of regions was actually greater than the number produced by a brightness partition. Therefore, we concluded that in the absence of semantic information, a partition based on brightness was superior to one based solely on hue.

4.4.2b Semantic Classification

One might expect to improve color partitions in a given domain by selecting quantization intervals corresponding to characteristic colors of the prominent objects. Recall that Lieberman had obtained reasonable partitions using just seven carefully selected classes of hue. These results could not be replicated in our landscape scenes because of the overlapping hue

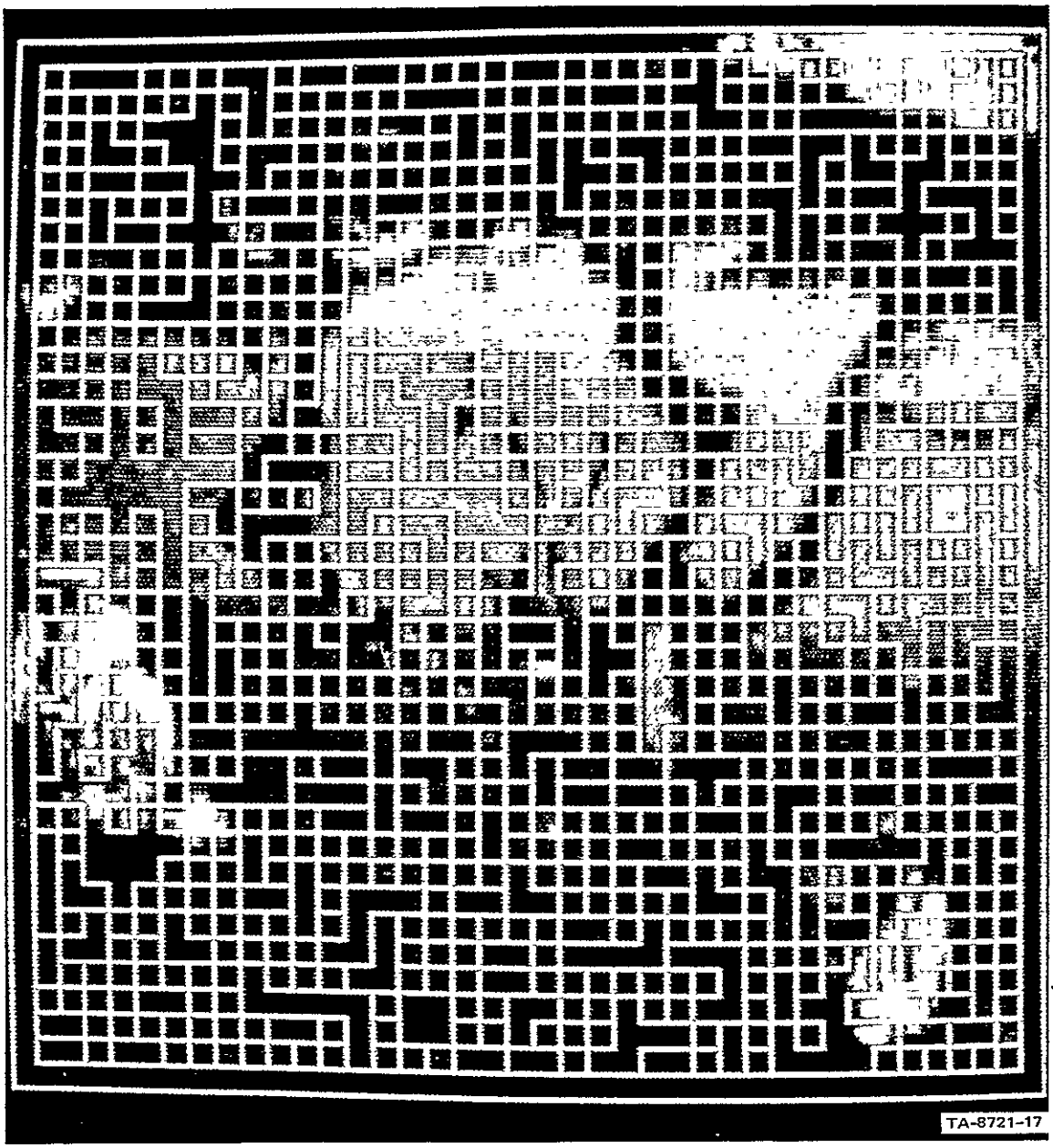


FIGURE 15 ' FIRST PARTITION OF LANDSCAPE PRODUCED FROM SAMPLES QUANTIZED INTO 20 DEGREE HUE INTERVALS (918 REGIONS)

distributions of the principal objects. Results in room scenes were also unsatisfactory, though for different reasons. While some interior surfaces have broadly distributed hues (e.g., patterned rugs, pictures, shiny tabletops, etc.), the majority have sharply defined hues. In color coordinated rooms, these hues tend to cluster in a very narrow range, decreasing the reliability of classification and increasing the attendant risk of an erroneous merge.

Although the overall partitions generated with semantic color classifications were unacceptable, particular regions with distinctive hues could be reliably extracted. For example, the orange hue of the chair appearing in room scenes supplied by Reddy and Ohlander is unique in those scenes (see Figure 1d). Consequently, image samples from the seat and back of that chair could be grouped into regions without risk of false merges. The number of regions that can be extracted in this way increases significantly when classification is based on multiple attributes rather than just color. Sky, for example, is the only region unsaturated in our landscape scenes brighter than 30 (on a scale of 31). Tabletop is the only horizontal surface higher than two feet found in SRI office scenes. These observations prompted a change in emphasis; rather than attempt to use range and color data directly in the partitioning process, these attributes would instead be used in a preliminary phase to extract regions composed of samples with clear semantic interpretations. Remaining areas of the scene (e.g., those containing semantically ambiguous samples) could then be partitioned conservatively based on brightness.

A number of experiments were

performed wherein samples of distinguishing objects were identified and grouped into regions prior to general partitioning. A set of predicates were empirically developed for selecting samples of each distinguished object. These predicates are similar to predicates used for filtering in object acquisition (see Section 5). Predicates were applied sequentially to all unclassified image samples. These samples passing a predicate were assigned a corresponding semantic label and removed from further consideration. Samples failing all predicates received a nonsemantic label based on brightness. The scene was then partitioned in the conventional way by combining into regions adjoining samples with the same labels.

The two most successful results are documented in Figures 16 and 17. These results should be compared with the corresponding brightness based partitions shown in Figures 18 and 19. Tables 4.1 and 4.2 document the selective classification criteria used. The following points are pertinent:

(1) In sequential classification, later predicates may be able to take advantage of context reductions achieved by earlier predicates. For example, the picture on the wall in Figure 1-d is a complex pattern. However, it was removed by a simple height criterion, once the wall samples had been accounted for.

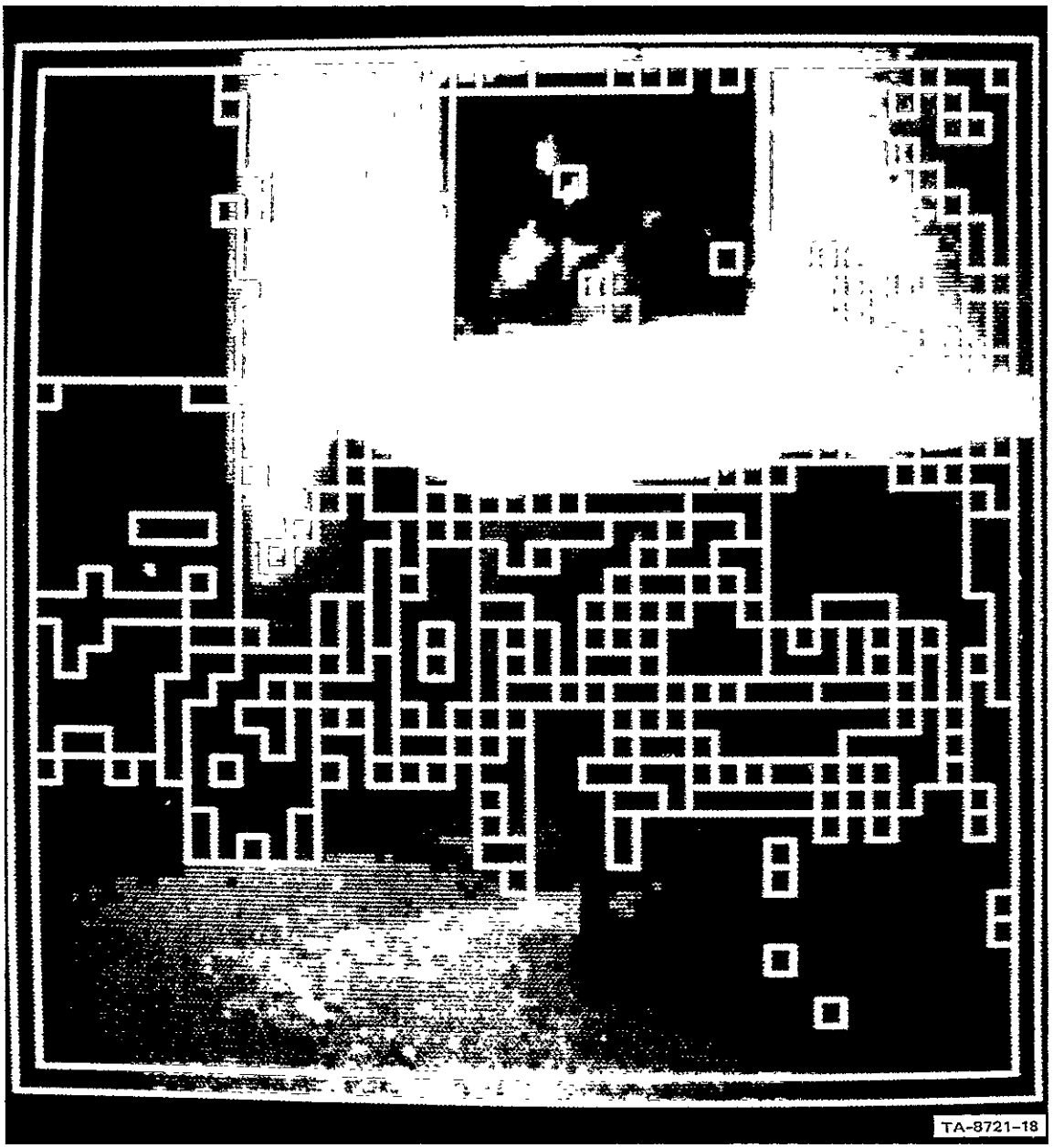


FIGURE 16 SEMANTIC FIRST PARTITION OF SRI OFFICE SCENE (235 REGIONS)

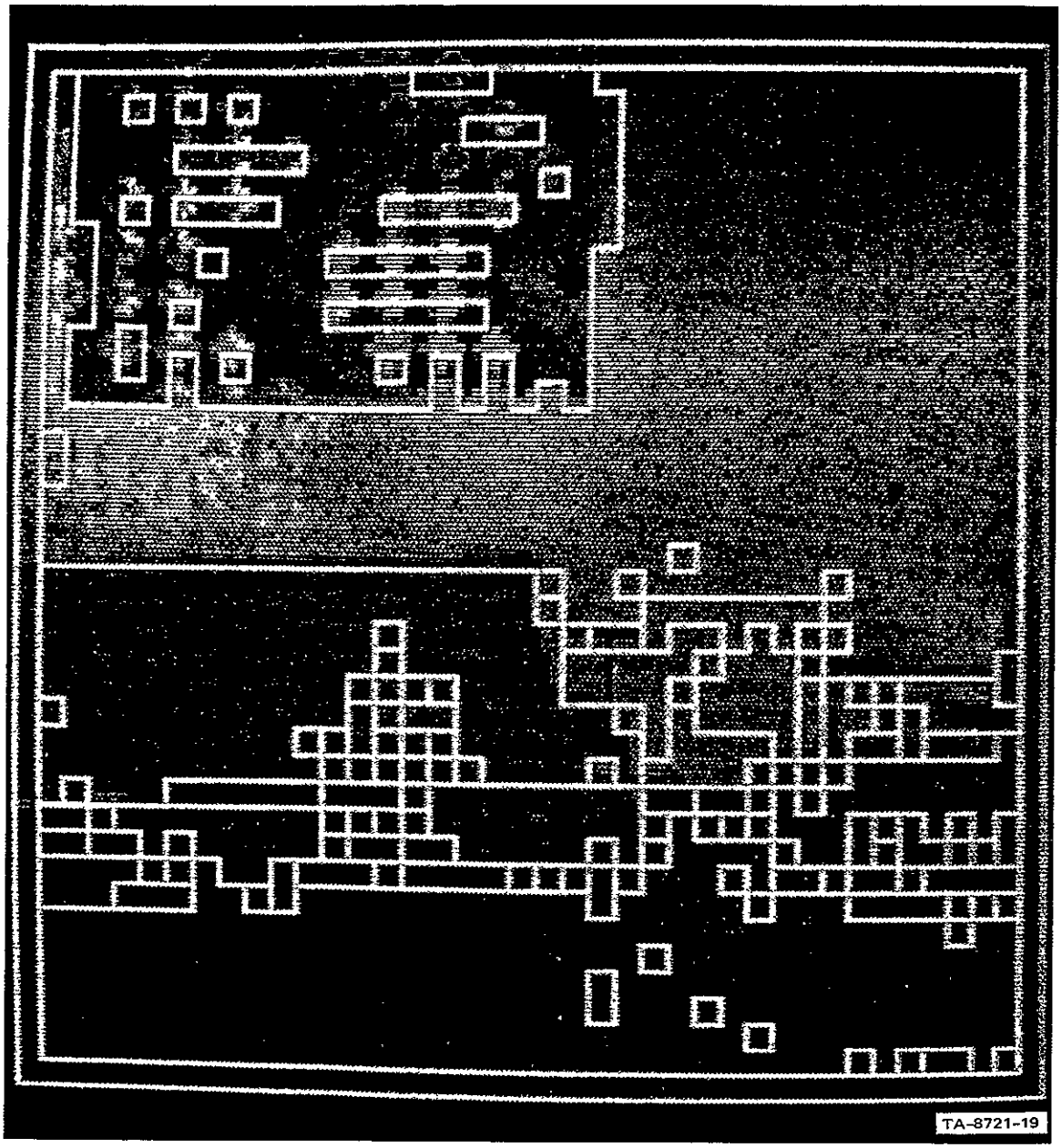


FIGURE 17 SEMANTIC FIRST PARTITION OF CARNEGIE-MELLON OFFICE SCENE (151 REGIONS)

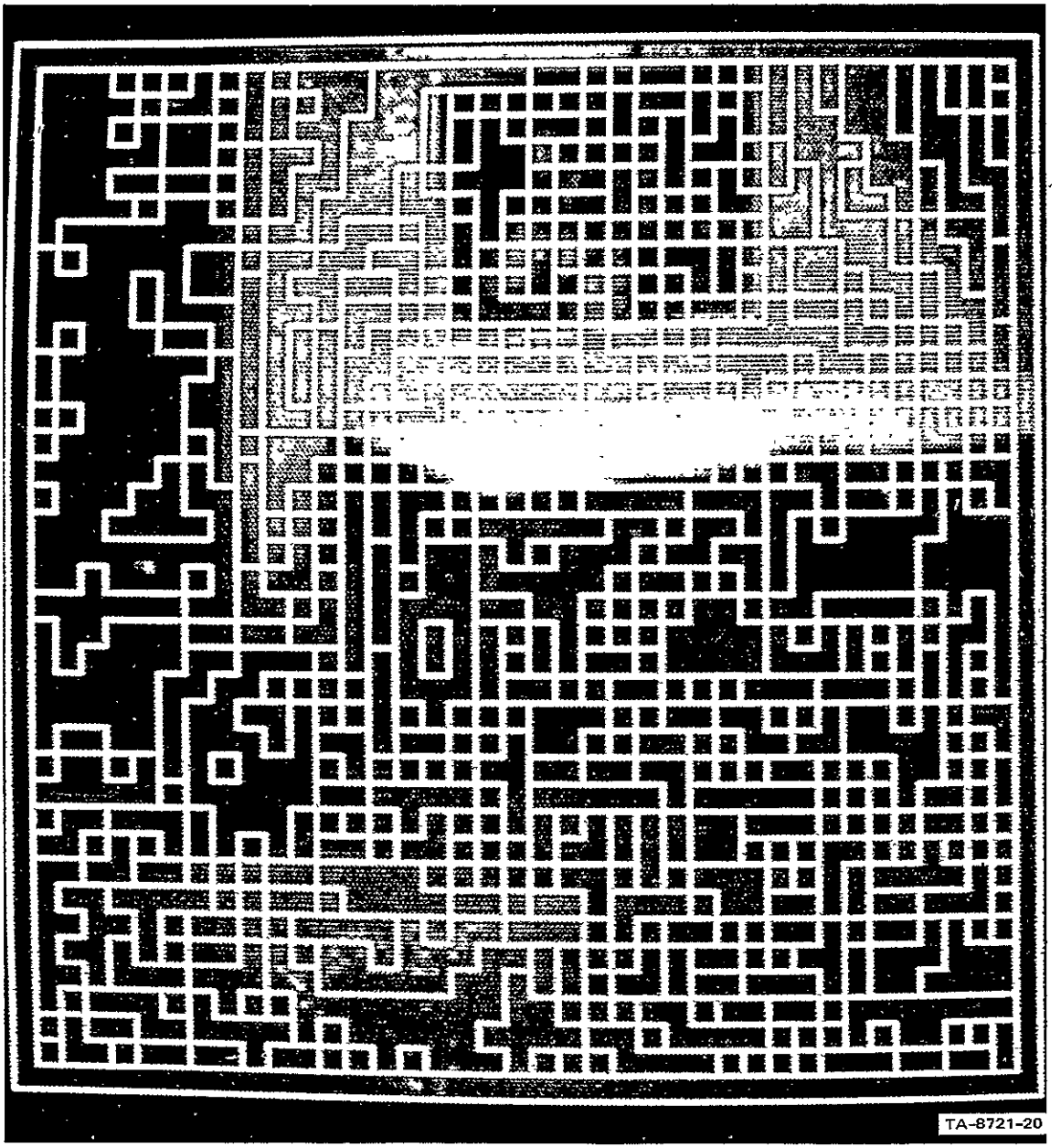


FIGURE 18 NONSEMANTIC BRIGHTNESS PARTITION OF SRI OFFICE SCENE (583 REGIONS)

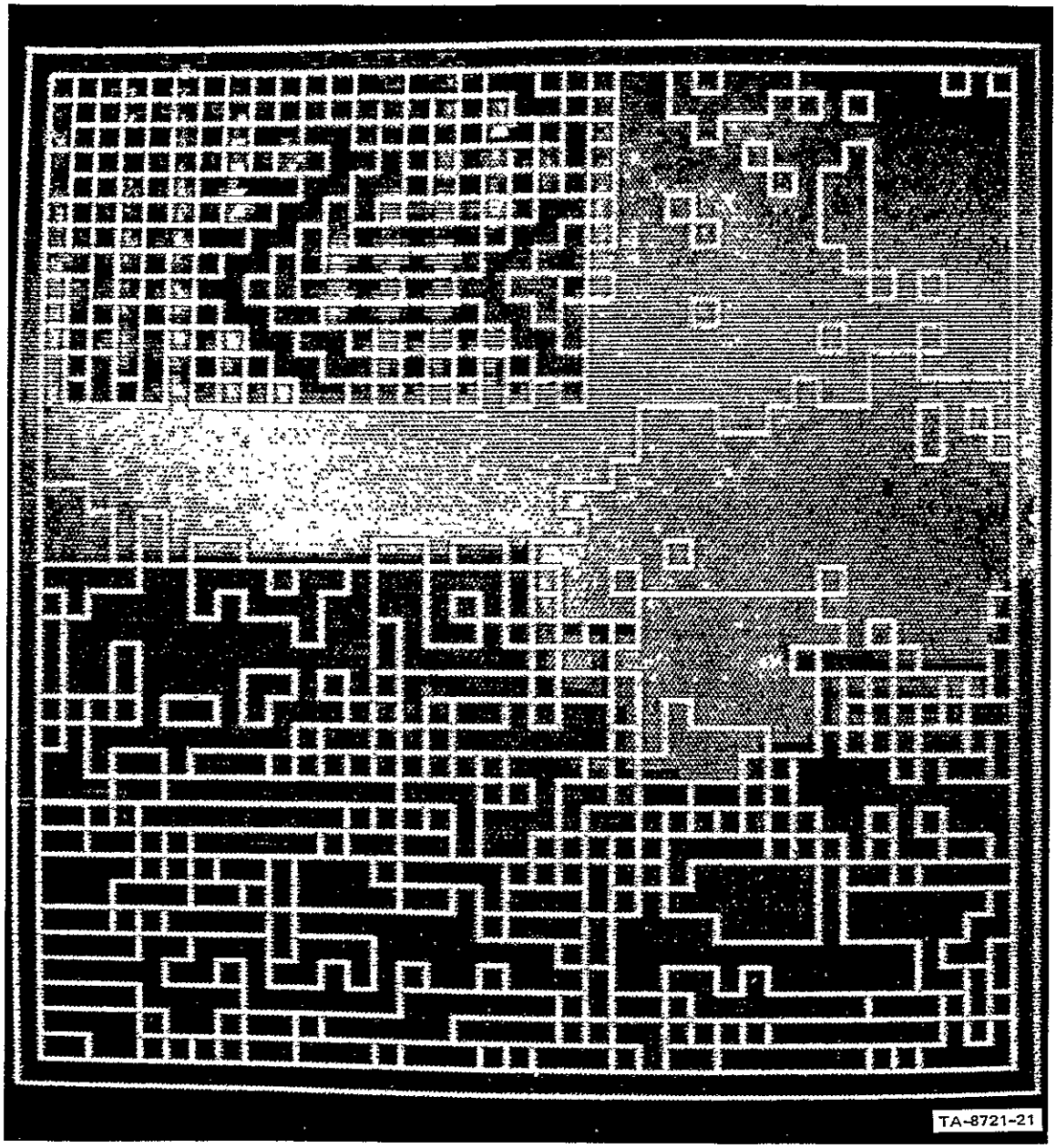


FIGURE 19 NONSEMANTIC BRIGHTNESS PARTITION OF CARNEGIE-MELLON OFFICE SCENE (441 REGIONS)

(2) Absolute classification is not essential; samples from different objects can satisfy the same classification rule without consequence, provided that those objects are not pictorially adjacent. Thus, no harm arises from the fact that regions of both door and picture in Figure 16 satisfy the fourth rule given in Table 4-1, since pictures in this domain are constrained to hang only on walls.

(3) Even when objects cannot be completely discriminated from their neighbors, it may still be possible to pull out substantial portions intact. The tabletop in Figure 17 was partitioned semantically into two characteristic dark regions while an ambiguous glossy area in the center was partitioned nonsemantically on the basis of brightness.

(4) Selective classification schemes are essentially limited to scenes where objects tend to be homogeneous and distinguishable by local attributes. Thus, while selective classification has given us good results for indoor scenes, it has proved to have little utility in outdoor scenes.

Given a suitable domain, it should be possible to use the acquisition planner described in Section 5 to derive the classification predicates automatically from pictorial examples. The classification task should, in fact, be simpler than global acquisition, since the predicate need discriminate only among pictorially adjacent surfaces.

Table 4.1

SELECTIVE CLASSIFICATION CRITERIA FOR FIGURE 16

1. Extract floor samples by height (<0.1 feet).
2. Extract chairseat samples by characteristic height and horizontal orientation.
3. Extract tabletop samples by characteristic height and horizontal orientation.
4. Extract picture samples in two passes.
   a. By characteristic height and saturation greater than maximum saturation for wall.
   b. By characteristic height and hue outside the hue range of wall.
5. Extract chairback samples by characteristic height, vertical orientation, and saturation.
6. Partition remaining samples by brightness.

Table 4.2

SELECTIVE CLASSIFICATION CRITERIA FOR FIGURE 17

1. Extract wall samples by their distinguished combination of hue, saturation, and brightness.
2. Extract picture samples by selecting all remaining samples in upper third of image.
3. Extract chair samples by their distinguished hue and saturation.
4. Extract couch samples by their distinguished hue and saturation.
5. Extract floor samples by their distinguished hue and saturation.
6. Extract tabletop samples by their distinguished hue, saturation, and brightness.
7. Partition remaining samples by brightness.

4.5 Merge Priority Experiments

The purpose of these experiments was to determine some fairly conservative nonsemantic measures of region similarity to use in defining the merge priority during region growing. It is important to note that at some point, all nonsemantic measures will commit errors, since they have no higher-level information about the scene domain. An example that illustrates this fact was suggested by Yakimovsky, who pointed out that in one part of a scene, it may be appropriate to merge small green and yellow regions comprising grass, while in another part of the scene, where yellow regions are part of a car, this merging is in error.

The nonsemantic merge priority criterion is intended to be used in conjunction with functions that apply semantic context to each proposed merge. The priority function should establish a reasonable merge order based on similarity measures, so that the semantic embedding can be performed on relatively well-defined portions of the scene.

The quality of nonsemantic similarity measures was compared by performing a first partition based on brightness, and then following a global best-first merge sequence using each similarity measure until there were only 250 regions remaining in the scene. The outcomes of the different proposed merge sequences were compared to see how well they honored the correct organization of the scene.

The first similarity measure tested was based only on

the brightness information obtained through a neutral density filter. For each boundary between two regions, the average absolute difference in brightness across the boundary was computed. This average is computed as follows:

(1) Average absolute brightness contrast $= \sum_{i=1}^{N} \frac{|br_{ai} - br_{bi}|}{N}$

where i ranges over the set of N adjacent picture element pairs along the boundary between regions a and b. br_{ai} is the brightness through a neutral density filter of picture elements in region A, and br_{bi} is the brightness of corresponding points in region b. The pair of regions with the smallest brightness contrast was merged first. The result is shown in Figure 20. Several serious, though very narrow leaks, occurred: the sea and ground were connected by a narrow neck to the left of the main tree trunk; the ground, sea, and horizontal tree branch were joined together to the right of the trunk, and the center vertical trunk became merged with the treetop. A major leak occurred between the main trunk and the ground, but this was not surprising, given the low contrast in that area.

Better results were obtained using a measure that was like the first, but that was based on the sum of the contrasts in each color separation. Thus, for each point on the common boundary between two regions, the absolute difference in r, g, and b was computed, and the average of this "color contrast" was computed for each pair of adjacent regions. The formula is:

(2) Average color boundary contrast $= \sum_{i=1}^{N} \frac{|r_{ai} - r_{bi}| + |g_{ai} - g_{bi}| + |b_{ai} - b_{bi}|}{N}$

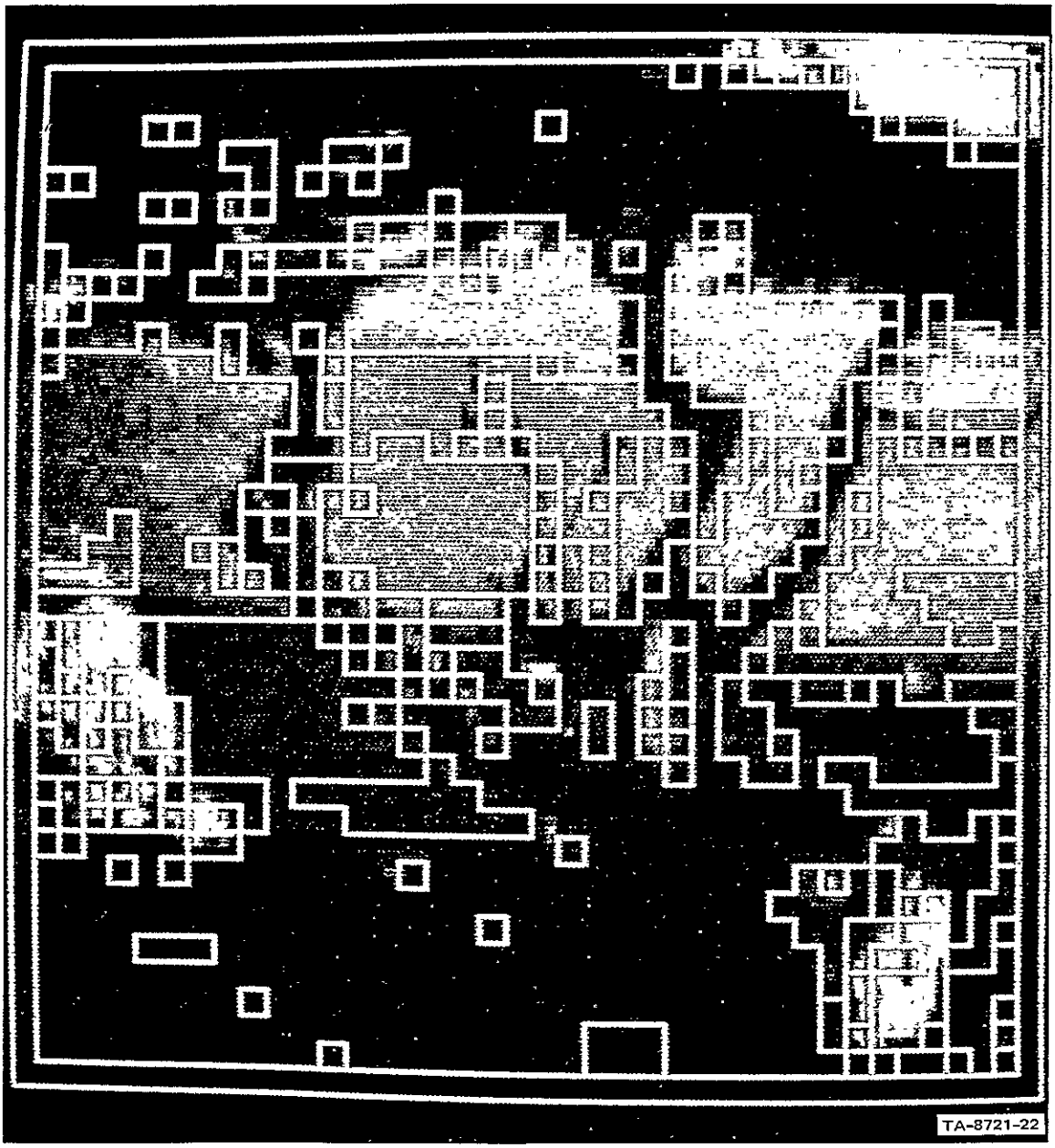


FIGURE 20 LANDSCAPE SCENE MERGED TO 250 REGIONS USING AVERAGE BRIGHTNESS CONTRAST ALONG THE BOUNDARY (EQUATION 1)

where r_{ai} , g_{ai} , and b_{ai} are the brightness of the ith boundary element from region a seen through the red, green, and blue filters respectively, and r_{bi} , g_{bi} , b_{bi} are corresponding data from region b, and i and N are defined as above. The result of merging regions with smallest color contrast first is shown in Figure 13. We also tried a similarity measure based on the sum of squared contrast differences, and obtained results that did not differ significantly from the results using absolute differences (see Figure 21).

Another similarity measure was to compute the average along the boundary between two adjacent regions of the maximum color contrast between any two picture elements drawn from sampling neighborhoods on opposite sides of the boundary. The attraction of this method is that it should behave very conservatively, declining to merge two regions if there is evidence in the full resolution picture that they are dissimilar. Unfortunately, this measure is too conservative, and noisy regions, like the ground and treetop, fail to coalesce before distinct smooth regions have grown together. The results are shown in Figure 22.

The similarity measures described so far are all based on boundary contrast properties. Another class of measures is based on the similarity of average region properties. On the basis of our success with the color contrast criterion discussed previously, we tried a similarity measure that used the absolute difference in average r, g, and b for each pair of adjacent regions. The formula for this similarity criterion is

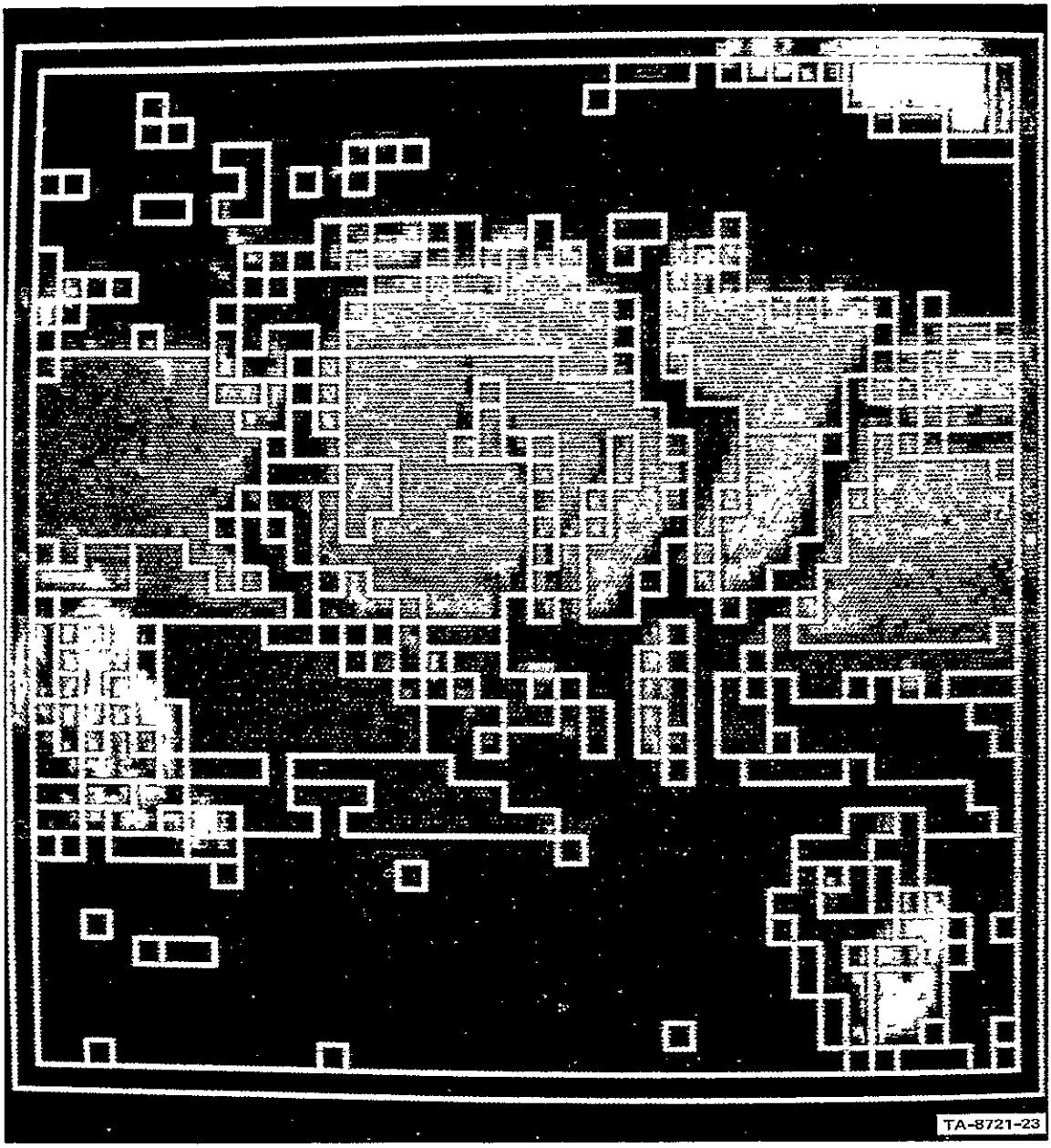


FIGURE 21 LANDSCAPE SCENE MERGED TO 250 REGIONS USING AVERAGE COLOR CONTRAST SQUARED ALONG THE BOUNDARY (EQUATION 2)

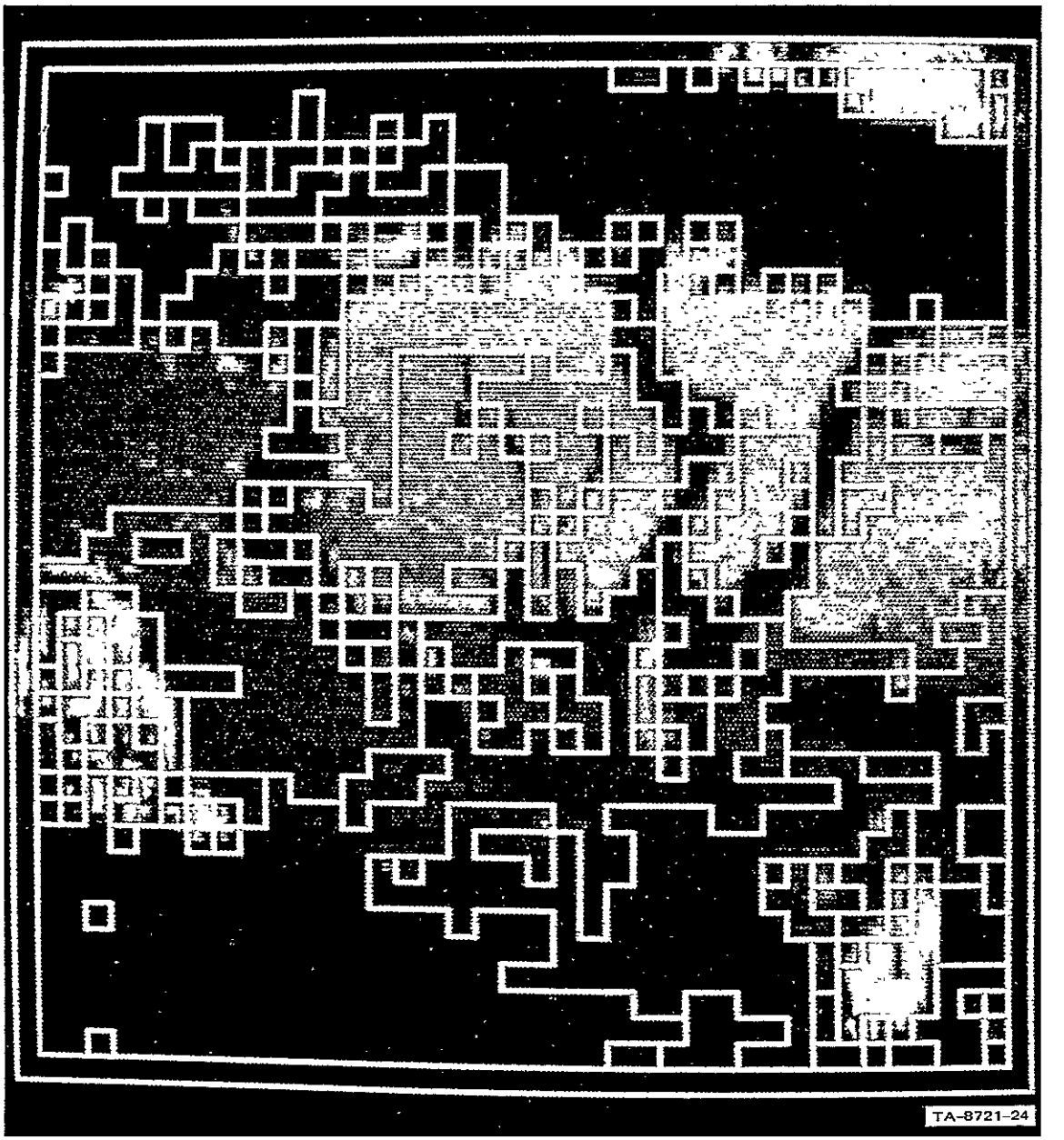


FIGURE 22 LANDSCAPE SCENE MERGED TO 250 REGIONS USING MAXIMUM COLOR CONTRAST ALONG THE BOUNDARY AT FULL RESOLUTION

(3) Average region color difference = $|\overline{r_a} - \overline{r_b}| + |\overline{g_a} - \overline{g_b}| + |\overline{b_a} - \overline{b_b}|$

where $\overline{r_a}$, $\overline{g_a}$, and $\overline{b_a}$ are the average brightnesses of region a seen through the red, green, and blue filters, and $\overline{r_b}$, $\overline{g_b}$, $\overline{b_b}$ and are the corresponding brightness of region b. The result, shown in Figure 23, is very close to and blue filters. The result, shown in Figure 23, is very close to the result using color boundary contrast.

The best results were obtained using a technique that combined our two best previous criteria. We defined the contrast between two regions to be the maximum of the color boundary contrast (Equation 2) and the average region color difference (Equation 3), and compared these maxima for each pair of adjacent regions. The results are shown in Figure 24. This was the similarity criterion used for subsequent experiments with semantics.

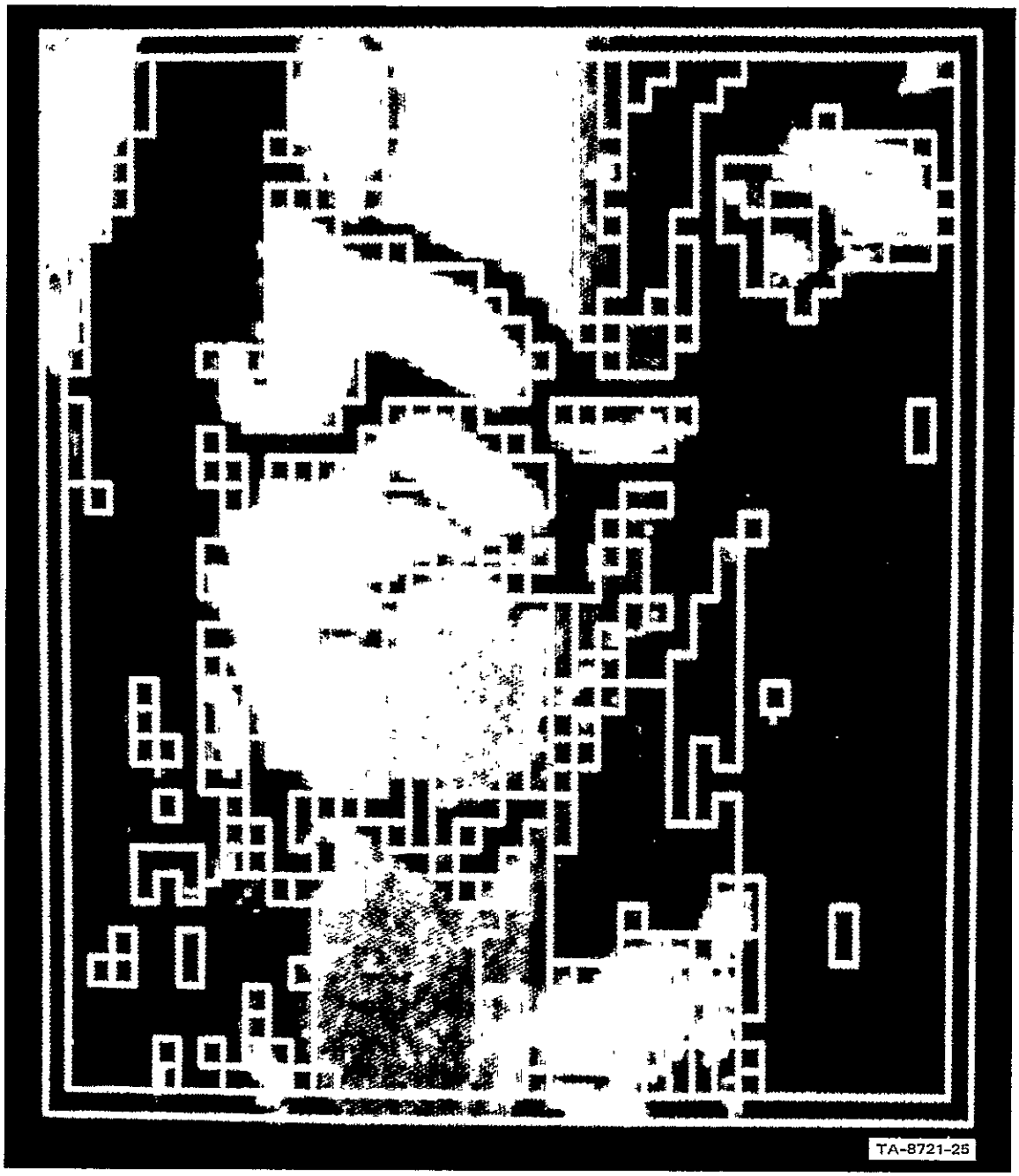


FIGURE 23 LANDSCAPE SCENE MERGED TO 250 REGIONS USING AVERAGE COLOR CONTRAST OVER THE REGIONS (EQUATION 3)

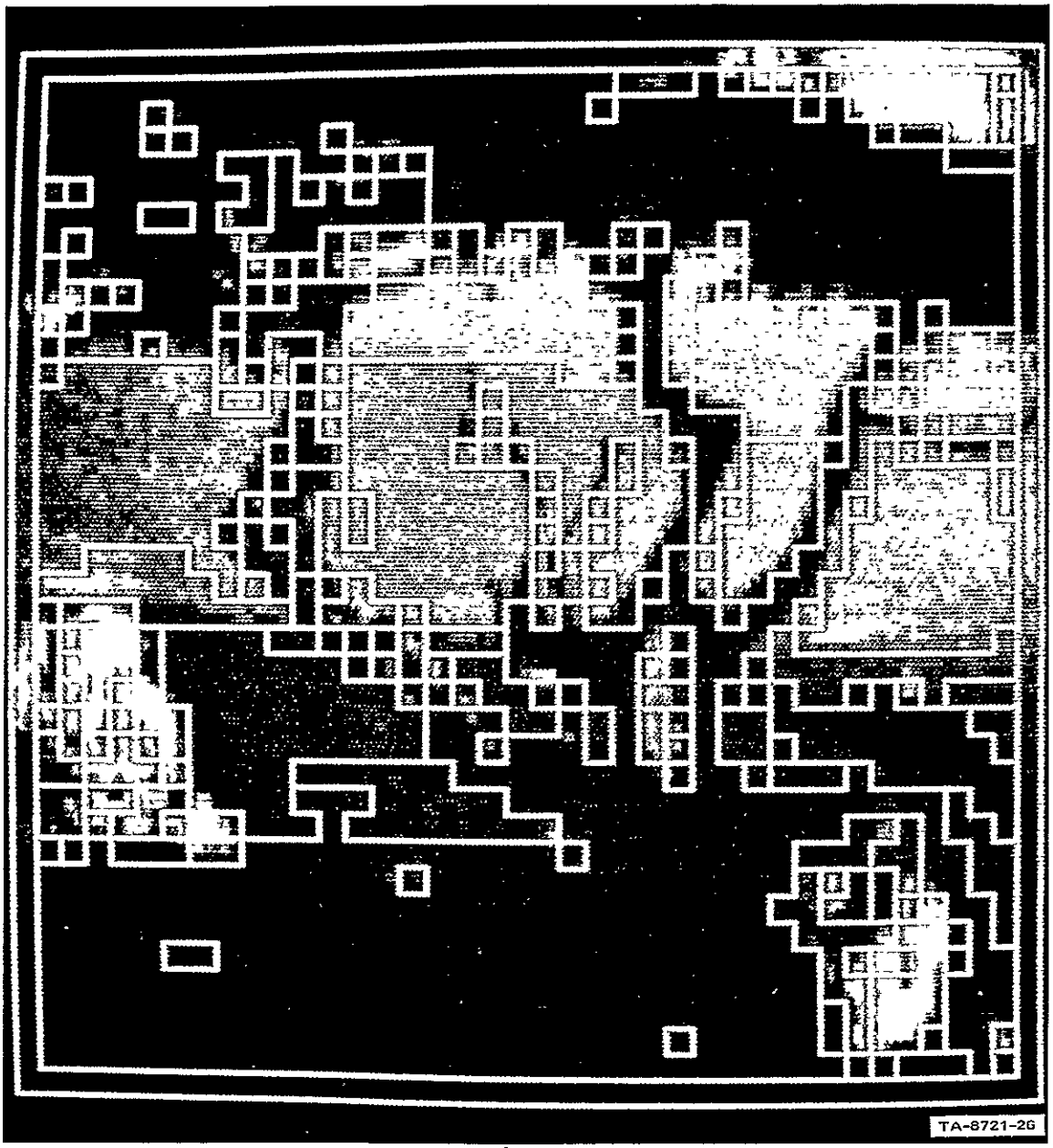


FIGURE 24 LANDSCAPE SCENE MERGED TO 250 REGIONS USING THE MAXIMUM OF THE AVERAGE COLOR CONTRAST COMPUTED ALONG THE BOUNDARY AND OVER THE REGIONS (EQUATIONS 2 AND 3)

4.6 Semantic Region Growth

We have seen in the region growing experiments above that regardless of the priority function, sooner or later an erroneous merge is proposed. Semantics can be used either to refine the boundary strength criteria so as to propose fewer erroneous merges [17] or else to block proposed merges that are incorrect. Stepping through merges proposed by the nonsemantic color difference criteria illustrated in Figure 24, it was observed that serious false merges seldom occur until the regions involved have grown sufficiently large to permit semantic interpretations based on region properties. This observation would suggest that merging errors could be avoided on semantic grounds simply by refusing to merge regions with different interpretations.

4.6.1 An Experiment in Semantic Region Growth

The above contention was tested interactively. The basic nonsemantic region growing algorithm was modified to check semantic compatibility before performing a proposed merge. Merges are approved only if both regions carry the same interpretation, or if at least one of the regions has not yet received an interpretation. Newly merged regions inherit the interpretation of their parents (or parent if only one carried an interpretation). When two uninterpreted regions are merged, if the size of the resultant region exceeds a threshold (in practice, the size of the smallest region typically involved in a significant erroneous merger), the program requests the experimenter to supply

manually a correct interpretation.

The modified program still runs primarily in a nonsemantic mode, merging regions across the currently weakest boundary, subject to semantic override. This process continues until all semantically compatible merges have been performed, terminating with a complete, semantically labeled partition of the scene. An indoor and an outdoor scene were each partitioned with only minor errors (due mainly to inadequate spatial sampling). The results are shown in Figures 25 and 26.

In both experiments, the size threshold for manual interpretation was set at seven samples. However, the great majority of regions inherited correct labels through merging before attaining that size. The final partition depicted in Figure 25 was based on the semantic first partition shown in Figure 16. Manual interpretations were provided initially for the 20 first partition regions (out of 235) that exceeded threshold size. Twenty additional interpretations were provided during the subsequent analysis when uninterpreted regions attained threshold size by merging. All major surfaces were extracted essentially intact except for chairlegs and tablelegs. These surfaces were too narrow for the available sampling density, and therefore corresponding semantic categories were omitted. The final partition in Figure 26 was based on an initial brightness partition (Figure 10). Although this first partition contained substantially more regions than the semantic partition used above, the number of (initial and total) manual interpretations required was surprisingly similar. In particular, 21

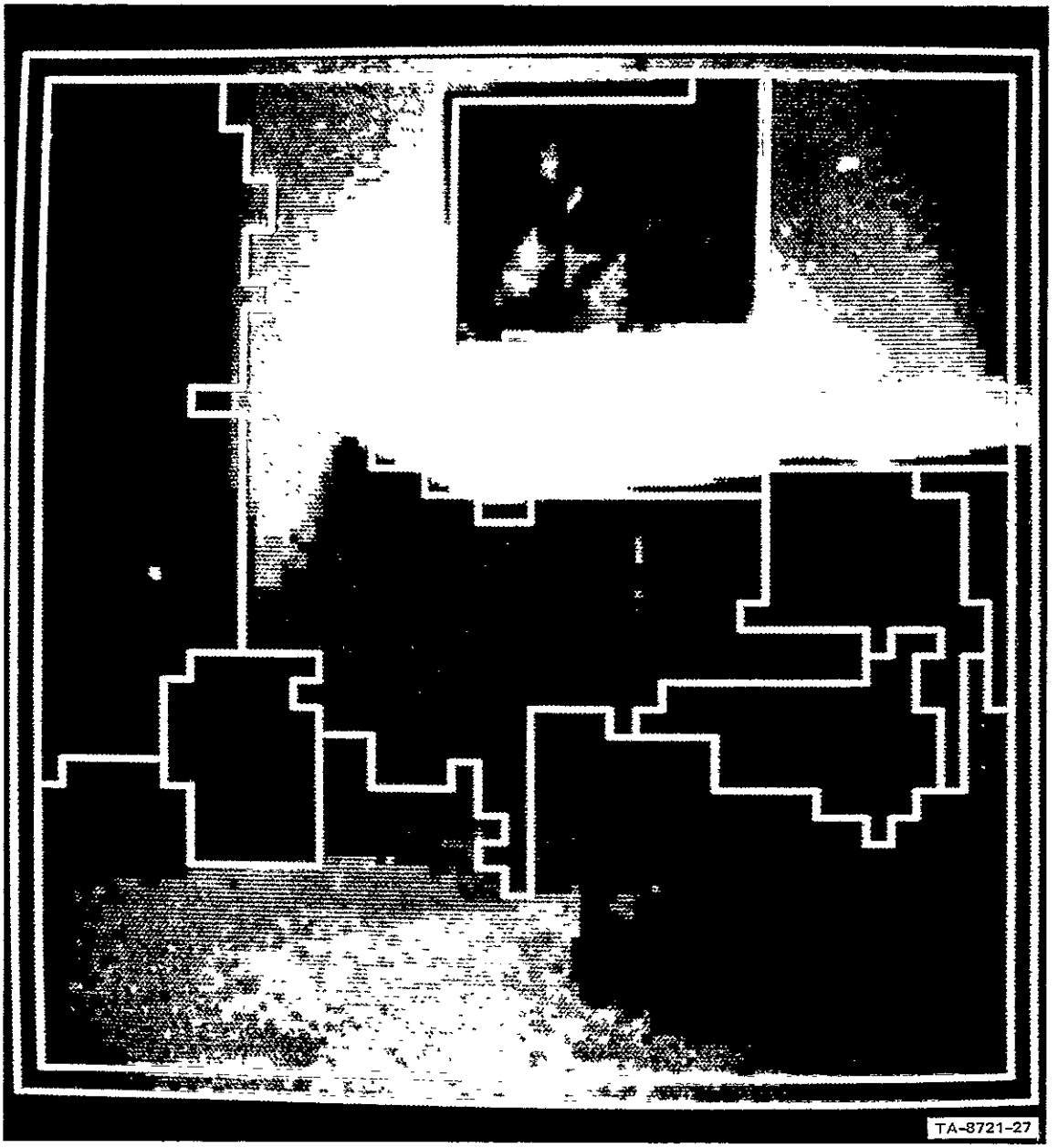


FIGURE 25 FINAL SEMANTIC PARTITIONING OF SRI OFFICE SCENE

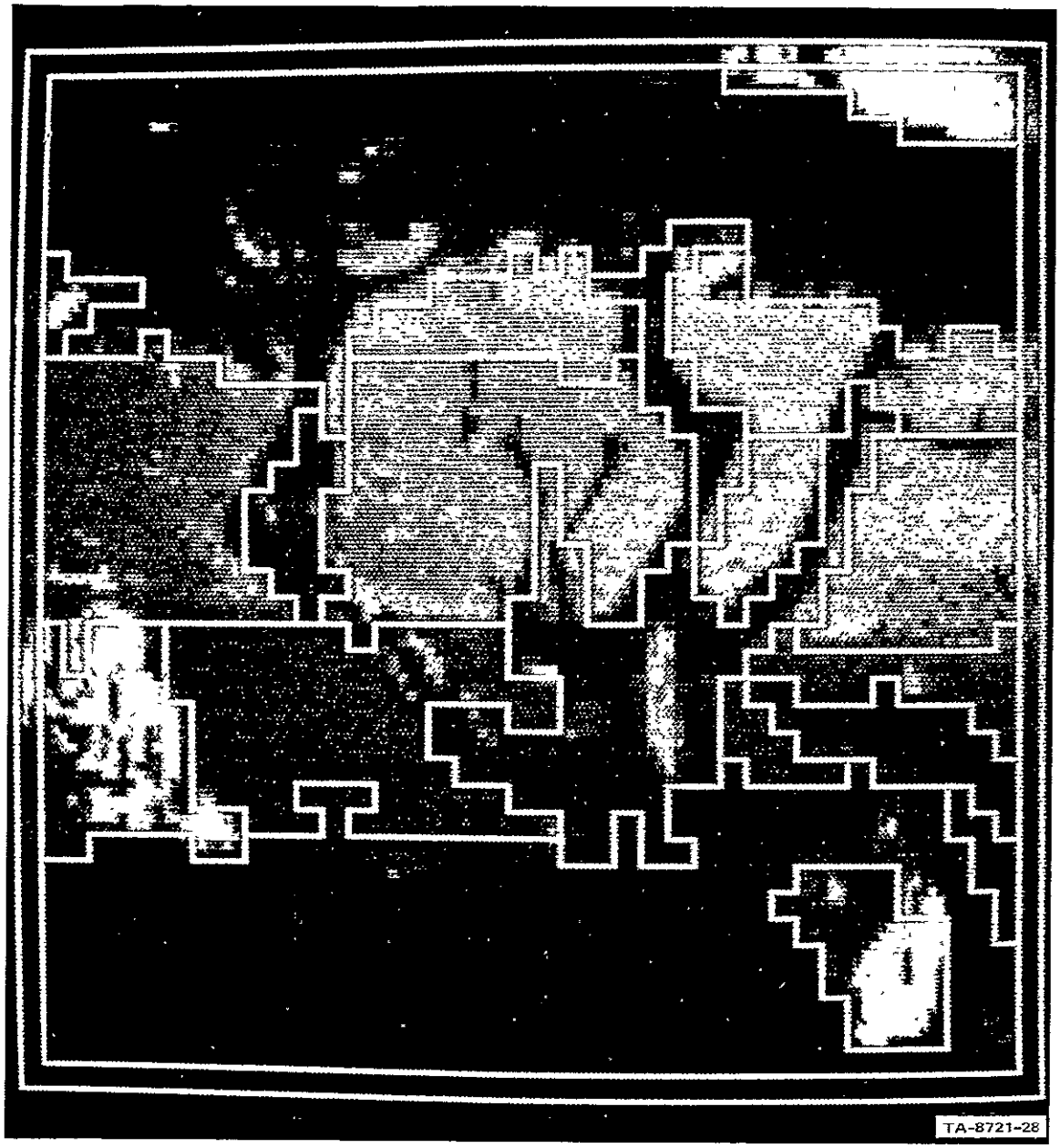


FIGURE 26 FINAL SEMANTIC PARTITIONING OF LANDSCAPE SCENE

first partition regions were larger than six elements (Figure 27) and received initial interpretations. The final partition (Figure 26) contains minor leaks between the sea and ground (in the lower right corner of scene) and between two fragments of sea across the left most tree limb. The sea-ground leak could have been avoided by setting the labeling threshold at six samples, the size of the sea fragment when the leak occurred. The sea-sea leak was a more fundamental error, resulting from a gradual erosion of small "tree bark" regions into the neighboring sea regions before a cohesive piece of "tree bark" could be identified.

The above results confirmed the original contentions that leakage occurred mainly between sizeable regions and that leakage could be minimized, provided that such regions could be semantically labeled. It was surprising how well a handful of manually introduced interpretations could improve on a completely nonsemantic scene partition.

4.6.2 An Experiment in Semantic Region Classification

The success of the above experiments suggested a second set of experiments aimed at studying the feasibility of assigning interpretations automatically to regions that attain threshold size. In the following experiments, regions were classified by comparing local attributes (e.g., hue and saturation distributions) with those of large training regions

designated by the experimenter.

Classifications were performed on each of the first partition regions that had received a manual interpretation in the experiment described in Section 4.6.1. In room scenes, most regions were uniquely distinguished by a conjunction of local attributes which included height and surface orientation (from simulated range data) in addition to hue and saturation. The large regions in Figure 16 were uniquely interpreted with the exception that the bottom of the door and the wastebasket could not be distinguished. In outdoor scenes, local discrimination (based only on hue, saturation, and brightness) proved much more difficult. In this case, automatic classification based on hue, saturation, and brightness eliminates only the grossly unacceptable interpretations, leaving many ambiguities to be resolved.

Table 4-3 summarizes the results of classifying regions in Figure 27 according to the training regions outlined in Figure 28. Each test region was compared to all training regions using a Kolmogorov-Smirnoff test on the corresponding distributions of brightness, hue, and saturation (see Figure 7). These comparisons establish for each test region three rank orderings of similar training regions. Training regions that appear in first or second place in at least two of these rank orderings are chosen as possible interpretations for the test region (presented in the third column of Table 4-3). A second type of classification was obtained using a Bayesian classifier described in Appendix A. Only interpretations with a likelihood greater than 10 percent of the

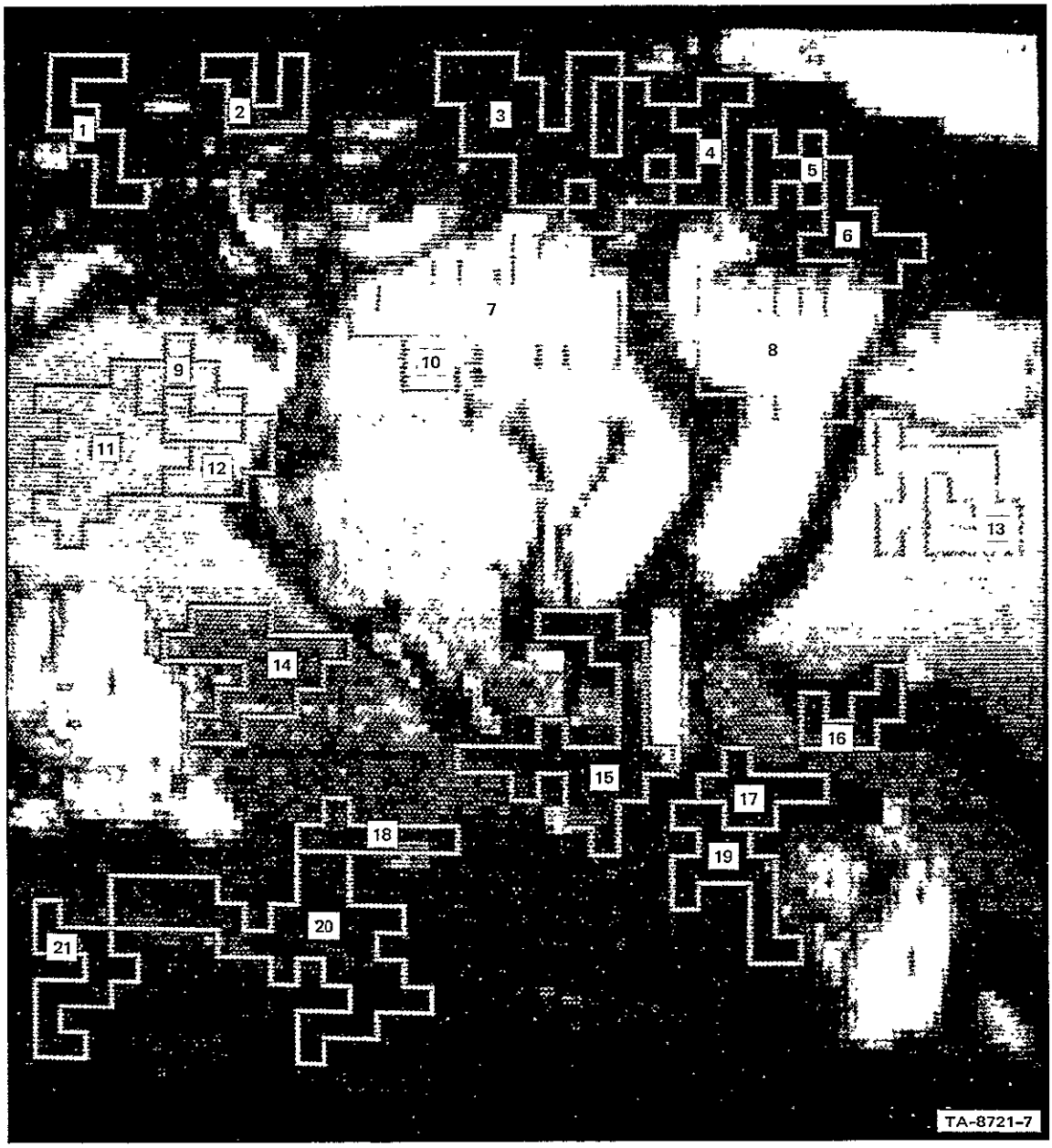


FIGURE 27 REGIONS FROM FIRST PARTITION OF LANDSCAPE SCENE (FIGURE 10) LARGER THAN 6 SAMPLES

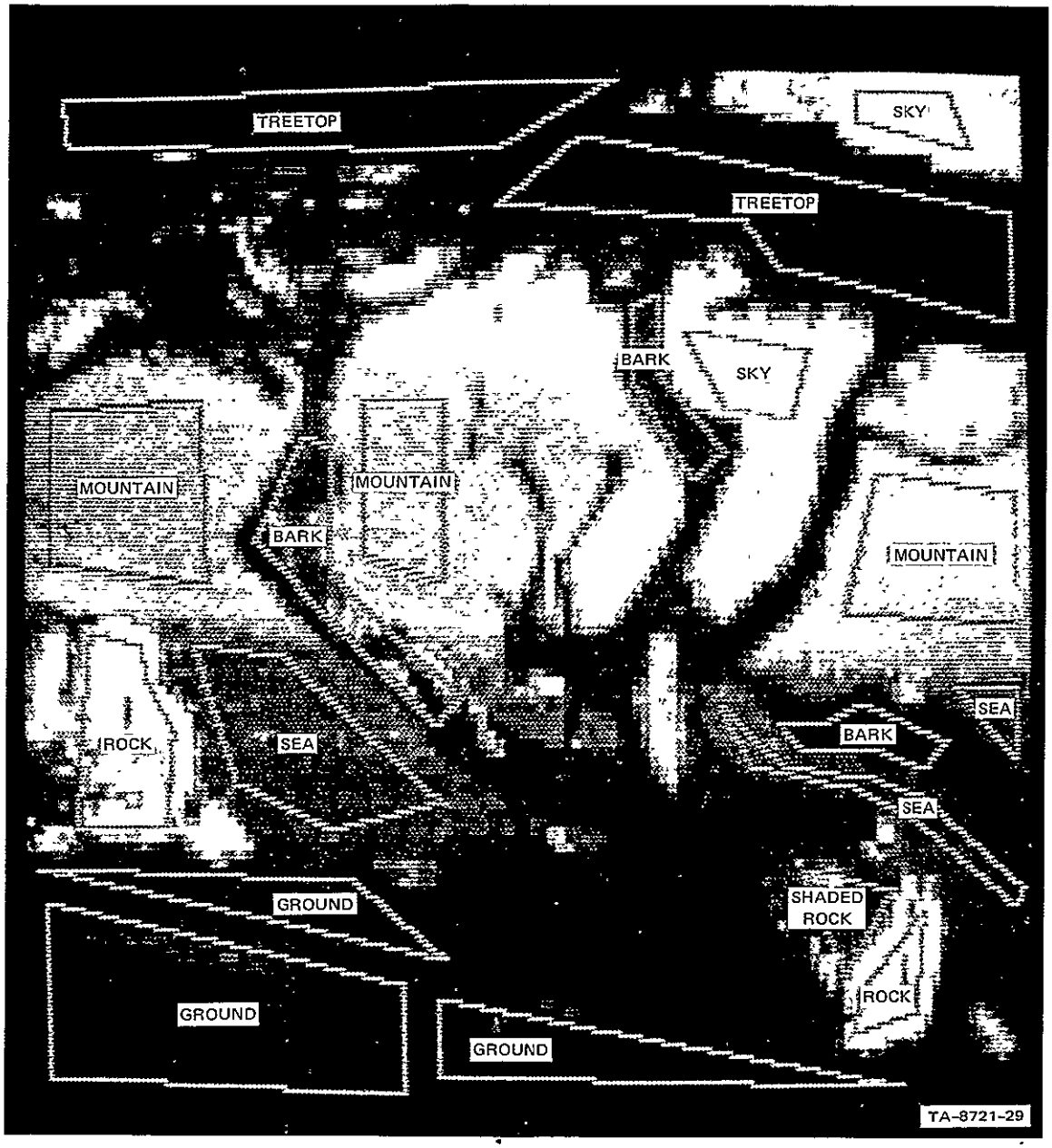


FIGURE 28 TRAINING REGIONS USED IN CLASSIFICATION EXPERIMENTS

likelihood of the most probable interpretation were retained. The results are shown in the fourth column of Table 4-3, with the Bayesian likelihood (between 0 and 1) in parentheses. The correct interpretation is not always the best match but is almost always among the top two or three alternatives. In several instances the interpretation ambiguities that occur are not serious because the alternative interpretations never appear pictorially adjacent. Note that these results are for training regions in the same scene as the test data. Worse results can be expected for distinct scenes chosen from a common domain.

Table 4.3

CLASSIFICATION OF REGIONS IN FIGURE 27
ACCORDING TO TRAINING REGIONS IN FIGURE 28

Region	Correct Interpretation	Majority ML Classifier	Baysian Classifier
1	Treetop	Treetop	Treetop(.236)
2	Treetop	Treetop	Ground(.228) Treetop(.2)
3	Treetop	Treetop	Ground(.208) Treetop(.202)
4	Treetop	Treetop	Ground(.223) Trunk(.154) Treetop(.133)
5	Treetop	Trunk Treetop	Ground(.21) Trunk(.205) Treetop(.15)
6	Treetop	Treetop	Treetop(.257)
7	Sky	Sky	Sky(.985)
8	Sky	Sky	Sky(1.0)
9	Mountain	Sea Mountain	Mountain (.539)
10	Mountain	Mountain	Mountain(.798)
11	Mountain	Mountain	Mountain(.969)
12	Mountain	Mountain	Mountain(.653)
13	Mountain	Mountain	Mountain(.863)
14	Sea	Sea	Sea(.502)
15	Trunk	Ground	Ground(.201) Treetop(.191) Trunk(.161)
16	Trunk	Trunk Treetop	Trunk(.246) Ground(.224)
17	Trunk	Treetop	Ground(.212) Trunk(.147)
18	Sea/Rock	Sea Mountain	Trunk(.337) Treetop(.252) Rock(.0921) Sea(.0862)
19	Trunk	Trunk	Trunk(.206) Ground(.205)
20	Ground	Ground	Ground(.205) Treetop(.196)
21	Ground	Treetop Ground	Ground(.219) Treetop(.196)

4.7 Toward Automatic Region Analysis

The classification experiments show that in order to automate semantic region growing, the basic paradigm must be augmented to handle regions with ambiguous interpretations. In this section we shall outline briefly some plans for how this might be accomplished.

In admitting regions with multiple interpretations, provision must be made (a) for determining the semantic compatibility of a proposed merge, (b) for determining the interpretation of a newly merged region, and (c) for ultimately resolving any ambiguities. Newly merged regions will acquire the set of possible interpretations formed by intersecting the intepretation sets of its parent regions. If this set is empty, the merger is obviously incompatible. Otherwise, compatibility depends on whether the interpretations left in the intersection set are ever pictorially adjacent in the domain of interest. This point is best illustrated by example. In Figure 1c two regions, each classified as "treetop or ground," can always be merged without risk of a leak because "treetop" and "ground" are never adjacent in the image. Thus, since these two regions adjoin, they must have the same interpretation; either both are "treetop" or both are "ground." On the other hand, two regions, each labeled "treetop or treebark," should not necessarily be merged because an error would occur if the actual identity of one region was "treetop" while the other turned out to be "treebark." In the latter case, a decision on the merger must be deferred until the individual region interpretations can be further

constrained.

Ambiguous region interpretations can be refined in basically two ways: by considering additional, global region attributes (e.g., texture and shape), and by considering contextual constraints imposed by the possible interpretations of nearby (primarily adjacent) regions. For instance, regions of "treebark" "ground," and "treetop," in Figure 1c have similar color and texture. However, "tree trunk" regions, unlike most regions of "ground" and "treetop," are predominantly thin and vertically elongated (e.g., high ratio of vertical perimeter/horizontal perimeter). Treebark regions are further distinguished by their extended vertical boundaries with their neighboring regions of "mountain" and "water". Such distinctions will be exploited by developing, with ISIS, ad hoc domain dependent representations for distinguishing among particular interpretation ambiguities.

The above ideas evolved from two related objectives: (a) we wanted a semantic region grower that could use the kind of ad hoc semantic constraints that are so easy to develop interactively in ISIS; and (b) we wanted a system that could be trained incrementally by introducing new semantics to block specific errors as they are observed. The proposed system uses two types of semantic rules; one set defines distinguishing features of particular interpretations, while the other defines which interpretations can be legally adjacent. Erroneous merges can occur if the regions involved are mislabeled, or if the system is unaware that the assigned interpretations may be adjacent. In the former case, a trainer could use ISIS to refine empirically the local and contextual features associated with recognizing the missing interpretation. In the latter case, he would merely provide an explicit constraint prohibiting the merge of regions with those assigned interpretations.

Our proposed merger semantics are closest in spirit to the structure rules advocated by Harlow and Eisenstat, which could be added incrementally to block specific leaks. New rules could, however, interact with existing rules in ways that are difficult to predict a priori. Yakimovsky's Bayesian semantics, on the other hand, are intrinsically interdependent and must be acquired over many trials using stochastic learning procedures. The probabilities are also not used directly to block merges but only to order the priority in which merges are proposed. It is consequently difficult to introduce semantic constraints in direct response to specific errors. There are other minor contrasts between our proposed system and Yakomovsky's. In our system, no distinction is drawn

between syntactic and semantic stages. Semantics are continuously introduced into the analysis as individual regions achieve a coherent size. We also do not subscribe to Yakimovsky's assumption that boundary and region semantics are independent. Rather, it would seem that boundary relationships may, on occassion, be the strongest clue to a region's identity, which in turn could improve semantic boundary strength estimates used in setting merging priority. Yakimovsky's system, however, is still the most successful automatic region analyzer implemented to date.

4.8 Toward Interactive Region Analysis

For the foreseeable future, a large number of scene domains will be too complex to process completely automatically. Some of these scenes are also too detailed to segment rapidly by hand. For such scenes, an interactive methodology like that used earlier in studying semantic region analysis may prove useful in its own right as an alternative to purely automatic or purely manual partitioning; the human time and effort in entering roughly 20 semantic labels are probably less than the time and effort required to outline each of the 50 regions in the final partition of Figure 26. However, when interactive region analysis is identified as an explicit goal, numerous improvements come immediately to mind. The experimenter can, with relatively little effort, crudely outline and label the major regions. This crude outline can be used directly as a good initial partition from which detailed boundaries can be rapidly grown. The outline can also provide training data for automatically classifying the few additional regions that might attain threshold size without inheriting an interpretation from the initial regions. (Note that for this purpose, generality of the classifier over several scenes is not even an issue.)

Two other schemes for interactive region analysis were tried experimentally and are reported here, for completeness. In the first experiment, the scene was conventionally partitioned into regions of homogeneous brightness; however, the global priority queue was not built. Instead, the user was asked to point at significant objects in the scene with a cursor. Each time he pointed, the

boundaries between the first partition region at that location and its neighbors were entered onto the priority queue. When the user had indicated what he felt were the major scene entities, the system began to merge regions nonsemantically across the weakest boundary currently on the queue. After each merge, the queue was updated to include additional boundaries between the new region and its neighbors. We expected that regions would grow out from each starting kernel, and since merging was still done on a "best first" basis, that regions would halt at object boundaries. This expectation was not realized in experiments on landscape scenes because of the similar coloring of many objects and the textural variations within objects. Spatial proximity thus became a primary factor in determining the kernel to which a given picture element would ultimately become attached. Still, this method might be useful for rapidly obtaining a crude partition, provided that: (a) the kernel regions are large and centrally located within each object, and (b) the merging process is terminated early enough (say, with 100 regions remaining). Users were aided in their choice of kernels by superimposing the actual first partition regions on the displayed scene.

In the second experiment, semantic labels were assigned to the selected kernels, and were subsequently used to block incompatible mergers. The resulting partitions were consistently better than the nonsemantic version. However, quality still depended critically on the number of kernels selected and on their placement. For example, leakage can be minimized by choosing a succession of kernels along opposite sides of a desired final

boundary and then assigning different semantic labels to kernels on each side. Selecting at least one kernel in each pictorially isolated patch of an object (e.g., fragments of sea isolated by tree limbs) allows merging to continue further toward a complete partition without committing errors.

Allowing a user to choose semantic kernels all at once consumes considerably less of his time than requiring him to stand by to supply interpretations throughout an analysis. On the other hand, it is difficult to know a priori which kernel regions will ultimately lead to good partitions. Consequently, partitions resulting from semantic kernels were poorer than those obtained when interpretations were requested during the analysis. A contributing cause is the fact that in the kernel scheme, the best merge candidate is selected from a restricted universe of regions directly descended from the original kernels, rather than the set of all first partition regions. The kernel scheme is thus much more likely to propose erroneous mergers early in the analysis, when this discrepancy is greatest. A user's cost-effectiveness might be optimized by some combination of these approaches wherein requests for interpretation can be minimized by supplying initial kernels.

The semantic kernel approach might prove more suited for partitioning a scene into two classes (i.e., figure and ground) than for obtaining a complete partitioning. We plan to investigate its use as an alternate means for inferring intent when a user points at an object.

5. AUTOMATING GENERATION OF OBJECT FINDING STRATEGIES*

5.1 Introduction

In this section we describe a system that can be rapidly programmed to find major surfaces in relatively complex real-world environments. Objects are designated to the system by circling examples with a cursor in a displayed image. The system formulates a strategy for finding the object, based on knowledge of available picture processing techniques and about the makeup of the current pictorial domain. The resulting program is then run on representative scenes and debugged interactively as errors materialize. This work is part of a system that can be rapidly programmed to find objects in office scenes as described in [1].

Previously we have used ISIS to develop interactively a set of procedures for finding objects in office scenes. The basic idea underlying these strategies is to rapidly disqualify obviously irrelevant areas of the scene by sampling for characteristic attributes of the desired object (also referred to as the target object or target)*. Additional local properties are then tested to eliminate those samples belonging to other objects that share the acquisition attributes. The surface containing the acquired samples is then bounded, and the resulting region is tested for appropriate size and shape. Programs of this sort have been developed for finding various objects commonly found in office scenes (including door, table, chair, wall, picture, and floor).

*Work reported in this section was supported in part by the Advanced Research Projects Agency under Contract DAHC04-72-C-0008.

A plan for finding a door might consist of acquiring samples at a height likely to contain few things besides a door, and then validating by eliminating those samples which, on the basis of remaining local attributes, have low likelihood of being part of a door. The door boundary is then obtained by growing a region around the initial samples, using attributes such as hue and saturation. One alternative approach would be to first scan downward from the top of the image to a point whose height is unique to door and wall. The edge of the door is then obtained by scanning horizontally from this point, seeking an abrupt change in saturation. (Doors in SRI offices are deep brown while walls are tan.) If this edge has sufficient vertical extent, the door boundary is then inferred from the known shape and size of our office doors, and the local surface orientation measured in the image on the door's side of the discontinuity. Finally, the predicted boundaries are confirmed by testing for evidence of edges in the image.

The above strategies can be schematized into acquisition, validation, and bounding phases. Specific techniques for accomplishing each phase are chosen from available methods on the basis of current information about the pictorial domain.

5.1.1 Acquisition Methods

The most straightforward way of acquiring

samples of the object is to filter random picture samples, rejecting any sample that fails to pass a predicate descriptive of the desired object. For example, the sampled scene may be filtered to retain only those samples at a certain height. Alternatively, the scene can be sampled along a locus to detect distinguishing boundary discontinuities. These predicates are generated automatically from pictorial examples.

5.1.2 Validation Methods

Acquired samples are validated by checking additional local characteristics that distinguish the desired object from others with common acquisition attributes. These additional attributes are examined sequentially, reducing the number of samples that must be examined by successive tests. Total cost is minimized by deferring computationally expensive procedures. When the cost of planning the optimal order exceeds the known cost of using all remaining attributes, samples are classified with respect to possible objects. Those samples judged likely to belong to the desired object are retained. Samples along a boundary are further validated by testing the boundary length, strength, and orientation using suitable masks.

5.1.3 Bounding Methods

The final bounding phase is accomplished in various ways. Given a suitable geometric model of the object and a projective camera transform, a procedure can compute analytically the

best global boundary consistent with detected edge points. The computed boundary can be projected onto the image and validated using extent masks as above. Procedural models are available for common surface types such as horizontal and vertical rectangles. In the absence of such models, a region may be grown around validated points; neighbors of validated samples which pass a chosen predicate (often the acquisition predicate) are added to a push-down stack. Neighbors of points on this stack are then examined recursively. All accepted points are collected together into a region, for which a crude boundary can be obtained by computing a convex hull.

5.2 Automatic Generation of Predicates

The work described here is an attempt to automate three key decisions involved in planning a distinguishing features strategy:

(1) What attributes should be used for acquiring initial samples?

(2) What attributes should be used for validating those samples?

(3) What attributes should be used in region growing to obtain a complete outline?

All three decisions involve distinguishing a desired surface in a context of other surfaces that may be present. Acquisition, for instance, involves distinguishing a surface from other known surfaces in a domain, validation involves distinguishing a surface from other surfaces known to satisfy the acquisition attributes, and bounding involves distinguishing a surface from all surfaces that could possibly be adjacent to it in the image.

In order to explain predicate generation, a digression to describe the data base is required. Objects are characterized by local surface attributes, symbolic properties, and relations with other objects. The local surface attributes that are currently used as descriptors are: brightness, color (separated into hue and saturation components), and when range data are available, height and local surface orientation (referred to as "orientation"). Local properties are stored as cumulative histograms representing the probability density functions for the object's attribute values. These histograms allow the probability of an object having an attribute in a given range to be computed by a simple subtraction. These histograms may be augmented by symbolic properties such as boundary descriptions, size and extent values, surface characteristics (e.g., horizontal), and spatial relations (e.g., adjacent to, above, or left of). The creation and modification of these descriptions are described below.

A detector (or a "simple detector") is a procedure that checks whether a sample's attribute value lies within a given contiguous range of attribute values. A "composite detector" is a

conjunction of simple detectors. A "predicate" is the corresponding LISP program for a detector. A predicate returns T (true) if the value is within the range, and NIL (false) otherwise. The terms detector and predicate will often be used synonomously.

5.2.1 Design of Acquisition Predicates

Filtering randomly selected samples with a predicate that accepts only those of interest has proved to be an extremely useful acquisition tool in manual strategies. Acquisition predicates should pass a few points on the desired object while excluding all others. They should be as cheap and reliable as possible. Naturally, these requirements are somewhat contradictory; usually a detector that allows most points on the target object to pass will also pass some points from other objects. To overcome this difficulty, the program attempts to generate compound detectors that exclude all samples from undesired objects, allowing at least a few points on the desired object to pass. The program returns a list of any undesired objects that cannot be completely excluded, so that these may be distinguished subsequent to filtering, using more expensive global attributes.

The expected cost and confidence of applying a detector can be computed for each object. The confidence of an object given a detector is loosely defined as the probability that a sample accepted by the detector belongs to the object in question. The complete derivation is contained in Appendix A. Cost is defined as the expected cost (measured in milliseconds of CPU time) of application of the detector to the image. Given a detector and the expected number of points from the set to be filtered that will both be part of the object and be accepted by the detector, it is possible to compute the sampling frequency and the anticipated cost.

This derivation is also contained in Appendix A.

The quality of a detector is evaluated using a heuristic function of cost and confidence that reflects the design priorities listed above. With this detector quality function, Q (described below), the generation of good compound detectors becomes (for combinatorial reasons) a problem in heuristic search. The search proceeds as follows. First a good initial set of simple detectors is selected. The candidate detector with the best Q is then combined with others to see if the combination improves the Q function. The best combination, in turn, is combined with remaining candidates, and so on, until a terminating condition is reached: either the set of candidates is exhausted, some combination exceeds a preset Q threshold, or some effort bound is exceeded.

The heuristic quality function, Q, which is used to order a set of detectors, is shown graphically in Figure 29. Its maximum value is the confidence of the object and detector (which cannot be greater than 1.0), and its minimum value is 0. The cost of applying the detector determines where the detector falls on the curve. For equal confidence, and costs below a certain minimum, MINCST (currently equal to 2 seconds), the Q function is indifferent as to which detector is selected. For costs above a certain maximum, MAXCST (now set to 100 seconds), the function returns 0. In essence, any cost below MINCST is effectively zero and any cost above MAXCST is effectively infinite. In between, the curve is linear with a slope determined by the confidence, MINCST and MAXCST. By making the maximum value of Q be the confidence, added emphasis is

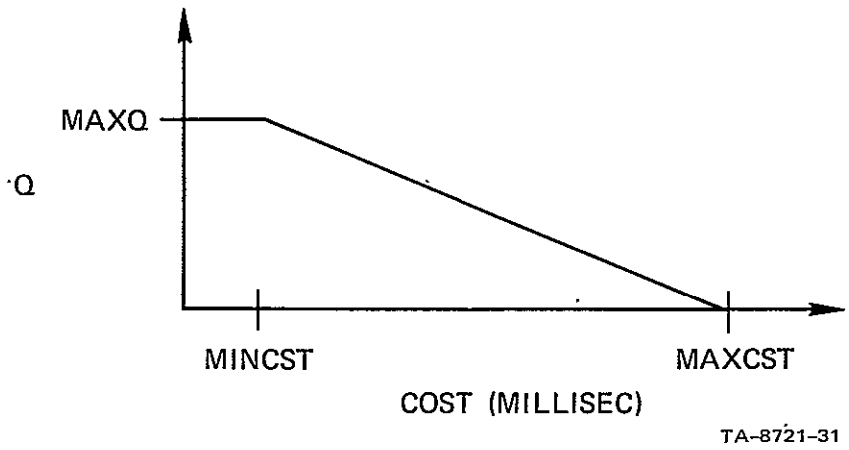


FIGURE 29 HEURISTIC QUALITY FUNCTION Q

given to the goal of producing high confidence tests. Of two competing tests, the one with lower confidence must also have lower costs in order to be considered better. Q was defined empirically to provide the functional characteristics detailed above. The parameter values were also chosen empirically on the basis of best results.

Two heuristics are used to select candidate simple detectors. The first uses the function, PEAKS, to locate peak attribute values for the object. PEAKS examines the characterization of the object (which are probability density curves and will be discussed later) for a peak providing a certain minimum area underneath. The function will then narrow the peak if it can do so without significantly decreasing the area. Minimum acceptable peak size and other variables are empirically determined parameters. PEAKS, which currently can only find a single peak for each attribute curve, outputs a set of simple detectors corresponding to the attributes considered.

The second selection program, MINRANGE, generates distinguishing detectors by looking for continuous ranges of attribute values which minimize the set of ambiguous objects. The program scans a range of an attribute for the target object, checking to see what other objects share the range. MINRANGE also returns a set of single detectors.

The present implementation of the heuristic predicate generator creates only composite detectors from the initial set supplied by PEAKS and MINRANGE. The program cannot yet generate

detectors that are disjunctions (such as might be required for locating a red and white checkered tablecloth) or decision trees. Despite these limitations (which will probably be removed in the future), the detectors generated are quite effective.

To generate these composite detectors, the program, DACQ (Default ACQuisition), selects the current best (in terms of Q) detector on the list and tries to combine it with all the simple detectors on the initial list. Detectors that would violate the description given above (e.g., which would combine two detectors with the same attribute), or those in which the composite does not have a smaller ambiguity set than the parents are not considered. When a candidate is selected, a composite detector is formed and retained only if its value of Q is higher than that of both parents. Before adding the new candidate to the appropriate spot in the list of detectors, a quick check is made to see if it could ever be better than the current best detector. This is done by computing Q for a hypothetical detector with the same cost, but with a confidence of 1. If the Q of this combination is higher than the best Q so far, the new candidate is kept. Otherwise, it is discarded.

A candidate that is retained is added to the proper location in the ordered list of candidates. After the first detector has been tried with all the possible single detectors, it is marked as having been expanded, and DACQ iterates with the new list. If no candidates are retained, DACQ continues with the next best detector on the list.

When there are no more candidates, or when a

detector is generated with a Q above a certain threshold, the program terminates, returning the complete ordered list of detectors generated.

DACQ spends considerable time attempting minor improvements to its best candidates. Therefore, a CPU time limit was provided to allow the user to decide how much effort DACQ should expend. If the time quantum provided by the user is exceeded, the program interrupts, prints the best candidate, and asks the user whether it should continue. If the reply is NO, or there is no reply within ten seconds (equivalent to a NO reply), DACQ terminates with the best so far. If the user wishes to proceed, DACQ will continue for another quantum of time. The predicates produced by DACQ within the time constraints (currently about 10 seconds) are typically the same as those produced by letting it run to completion. Also, the detectors produced are usually as good as, or better than, predicates defined manually by experienced users, and often they are the same.

The final steps in the automation of filtering are straightforward. The best compound detector is converted to an equivalent LISP LAMBDA expression, which is then compiled and saved. The system already knows what sampling density is required for the detector, since it is a by-product of the cost computation (see Appendix A). The image is then randomly sampled at the appropriate density, and the resulting samples are filtered with the LISP predicate to obtain a few acquisition samples. These samples may need further validation tests to discriminate residual ambiguities. Examples are found in section 5.3.

5.2.2 Design of Bounding Predicates

The next step in object finding after validating the acquisition samples is to extract the boundary of the surrounding surface. Bounding a single specific object is a significantly different task than the global scene partitioning described in Section IV of this report.

In object bounding, a region is grown from acquisition samples by including all contiguous samples that satisfy an appropriate bounding predicate. Specifically, the program, GROW, applies the predicate to samples adjacent to the acquisition samples. If a sample is accepted, its neighbors (either 4-adjacent or 8-adjacent) are added to a list to be recursively examined. When all possible samples have accreted to the original set, GROW returns the resulting new region.

The bounding predicates differ from acquisition predicates in two important ways. First, the particular region being bounded need only be distinguished from pictorially adjacent objects, as opposed to all objects in the scene. Second, region growth requires predicates capable of collecting all points on the object.

The basic region grower can be speeded up by scanning out horizontally and vertically from the initial acquisition samples until a discontinuity is detected. Points located on the boundary are then joined into a new, larger region, and GROW is used

to fill in any holes. Scanners are good for quickly locating edges, but suffer by requiring that all points between the starting point and the desired edge point pass the predicate. A point that fails terminates the scan. The region grower is less directed, but as a result, can "go around" isolated points where the predicate fails. The combination of scanning and growing has proven very effective.

Since the predicate must accept all points on the object, the task of generating an initial predicate is greatly simplified--we need only create detectors based on the complete range for all attributes (a "full-range detector"). There is relatively little risk in using these wide range detectors, since they need only discriminate objects adjacent to the acquired object. However, if the system has specific knowledge about object adjacencies, the full-range predicate can be refined by conjoining it with negated full-range predicates for those adjacent objects.

A major shortcoming of the above approach is that the region growing predicates are applied uniformly to all points. Often, it is known that the color (say) of a region is darker at the bottom than at the top. Such information should be used to produce more selective predicates that respond differently at different points in the region. Predicates could also be designed which continuously change their expectation of what should come next, based on what has recently been seen.

Other shortcomings are due to poor predicate specification from the outset. This problem can arise when the

system receives an incomplete description of the object initially, perhaps not knowing enough about other objects in the environment, or often not knowing of good descriptors which, although obvious to a person, may not be obvious to the system. Remedies for these shortcomings will be discussed in an upcoming dissertation [25].

In the coming months, the semantic segmentation techniques described in Section IV will be studied as an alternative means of bounding objects. GROW functions strictly by growing regions outward from initial kernal points on the target object. The other segmentation system could be used to grow simultaneously from kernels on the target and on adjacent objects. Additional knowledge about relationships of the target and its neighbors could be used to obtain improved boundaries.

5.3 Examples

Figures 30 through 37 illustrate the object finding process using the image shown in Figure 1a. The objects to be located are the picture and the major surfaces of the chair, the seat and the back. These objects (along with others that appear in the image) were shown to the system in other images of the same scene. Descriptions were automatically generated, and from these, the system was able to create strategies for locating the objects. In the figures, the regions created in the course of locating the objects are generally shown as a boundary with a few points inside. The boundaries (in this case) are convex hulls created from the samples by ISIS in order to emphasize the region for the user. In the LISP predicates mentioned below, the function LIMITP is used as the primary detector function, its format is: (LIMITP attribute sample low high). LIMITP accepts samples whose attributes lie within the range from low to high.

The first example shows the system locating the picture. Figure 30 gives the result of filtering a random sampling of the image with the predicate [LAMBDA (X) (LIMITP HEIGHT X 3.19 4.76)]. This predicate keeps only samples at a height within the range of 3.19 feet to 4.76 feet. Since only points from the wall, door, and picture should be found at this height, the validation phase need only distinguish among these three objects. The results of validation are shown in Figure 31. The system then grows the region shown in Figure 32 with the predicate (LAMBDA (X) (AND (PICP X) (NOT

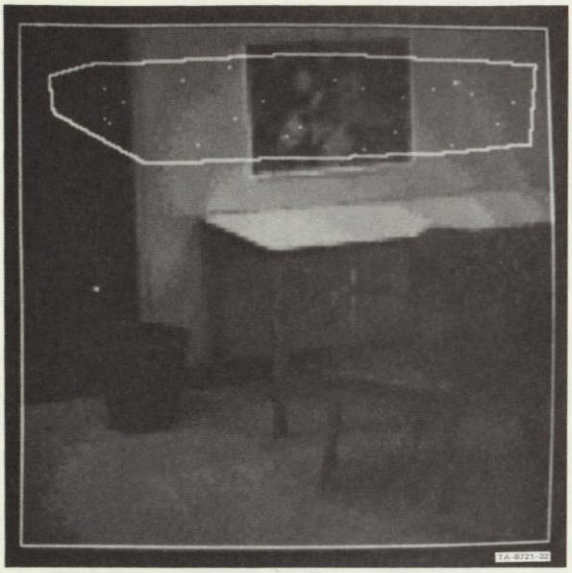


FIGURE 30 INITIAL ACQUISITION REGION FOR PICTURE

FIGURE 31 PICTURE POINTS AFTER VALIDATION

FIGURE 32 FINAL PICTURE REGION AFTER REGION GROWTH

(WALLP X)))). PICP * is a LISP predicate function, generated by the system which returns T if the sample has the properties of the picture. WALLP is a similar predicate for the wall. The complete growth predicate requires that the sample look like a picture point, but not like a wall point.

In Figures 33 through 37, the system locates the chair seat and back. In Figure 33, the sampled scene has been filtered for points on the chair back, using the predicate (LAMBDA (X) (AND (LIMITP HEIGHT X 1.62 2.14) (LIMITP HUE X 0.0 6.1))). This predicate, checking height and hue, selects only the single sample shown in the right middle of the chairback in Figure 33. The sample is retained during validation, and is used as a starting point for the region grower which generates and uses a predicate to discriminate the samples of the chairback from those of adjacent table and wall areas resulting in the region shown in Figure 34.

The acquisition, validation, and bounding process for the chair seat is shown in Figures 35 through 37. The region shown in Figure 35 is the result of filtering with the predicate (LAMBDA (X)

* These predicates are conjunctions of full range predicates for the objects in question. That is, they check that the value of each attribute of a sample falls within the possible range of values for the object in question.

FIGURE 33 SINGLE POINT ACQUIRED AND VALIDATED ON CHAIR BACK

FIGURE 34 REGION GROWN FOR CHAIR BACK

(AND (LIMITP HEIGHT X 1.09 1.62) (LIMITP ORIENT X 0.0 36.0))), which selects samples with height and orientation appropriate for the chair seat. One point is eliminated during validation, since its color does not match that of the seat, giving the region shown in Figure 36. Finally, Figure 37 shows the results of growing with a predicate that distinguishes the seat from the floor and the wall.

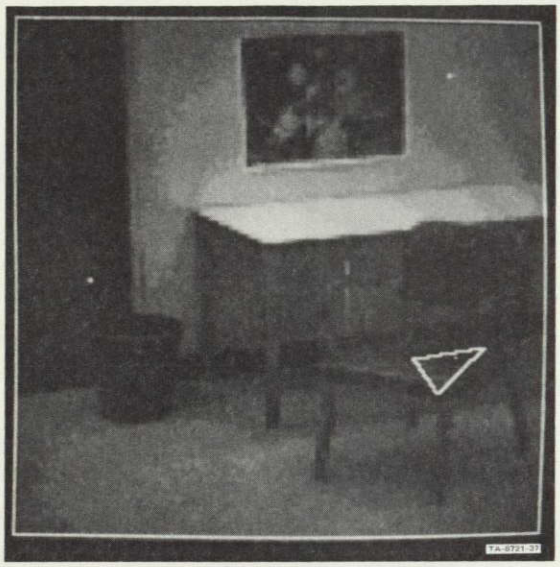


FIGURE 35 POINTS ACQUIRED ON CHAIR SEAT

FIGURE 36 POINTS REMAINING AFTER CHAIR SEAT VALIDATION

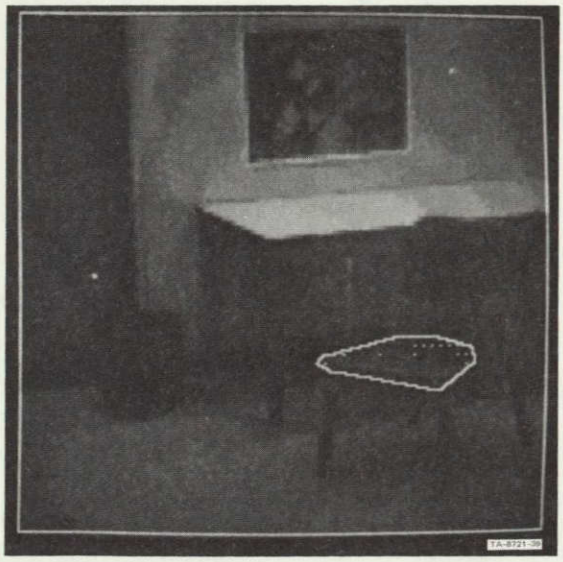


FIGURE 37 FINAL CHAIR SEAT REGION AFTER GROWTH

6. DISCUSSION

This report has discussed experiments in automatic and cooperative (man-machine) scene analysis, using interactive facilities of ISIS and associated subsystems. Significant results in the past year include: (1) development of techniques for cooperative scene analysis and their successful application to landscape scenes, (2) development of a new interactively trainable semantic region growing paradigm (based on 1), and (3) development of a program that can automatically generate strategies for finding or distinguishing objects, given pictorial examples.

The biggest questions regarding the utility of our interactive methodology concern its generality. Our results so far, are in-depth analyses of a few selected pictures from three domains. In the coming year, we will begin seriously to investigate generality on two fronts: (1) the applicability of strategies and semantics developed in one picture to other pictures in the same domain, and (2) the applicability of the basic methodology to additional domains, including several biomedical applications already underway.

The number of variables characterizing both scenes and scene analysis strategies present a big obstacle to systematic experimentation, and could explain why such experimentation has been generally avoided. To cite one example, the effectiveness of manually supplied semantic kernels on segmentation depends, among other things, on which regions are selected as kernels, on the order

in which they are chosen, and on the type of first partition being used. Experimentation is also relatively time consuming on a heavily loaded time shared system. For these reasons the experimental process must be automated to run on an annotated library of correctly analyzed pictures. Such a data base will allow experimenters to test the generality of proposed semantic descriptions in multiple scenes. It will be used to monitor the performance of automatic scene analysis strategies and alert the experimenter when an error is about to occur. The data base can even supply requested interpretations when testing interactive programs. The data base will be developed using interactive techniques to obtain perfect segmentations.

The continuing global objective of this project is to facilitate the process whereby a computer acquires the knowledge needed to analyze a set of scenes from a common domain. Our goals for the coming year are set forth in the following scenario: A user selects a representative scene from a new domain, crudely circles the principle regions on a display, and provides their interpretations. The system then completes the segmentation, perhaps requesting a few additional interpretations. The resulting partition is then used by the system to train classification algorithms and to deduce contextual constraints. Using this knowledge, the system next attempts automatically to partition additional scenes from the same domain. The user modifies the system's knowledge base when errors are observed.

The above scenario is based on clearly defined extensions of

present capabilities. Work has already begun on refining an initial crude outline into a complete partition (see Section 4.8). The semantic region growing paradigm will be automated in gradual steps along lines suggested in Section 4.7. Initially, regions reaching critical size will be automatically classified with respect to training regions. However, human assistance will be sought to resolve ambiguities. The second step will attempt to resolve ambiguities automatically, using additional, interactively formulated region attributes (e.g., shape and texture) and contextual constraints. An implemented constraint satisfaction system will continuously monitor the consistency of newly proposed region interpretations with those previously assigned, in order to resolve both current and pre-existing ambiguities [18]. The third and most tentative step will be to use the methods of Section 5 to deduce region descriptions and contextual constraints automatically from examples in a correctly segmented scene.

REFERENCES

1. J. M. Tenenbaum et al., "An Interactive Facility for Scene Analysis Research," Artificial Intelligence Center Technical Note 87, Stanford Research Institute, Menlo Park, California (January 1974).

2. J. M. Tenenbaum, "Accomodation in Computer Vision," Stanford A. I. Memo 134 (October 1970). Available from NTIS (AD 748 565).

3. A. R. Hanson and E. M. Riseman, "Preprocessing Cones: A Computational Structure for Scene Analysis," COINS T.R. C-7, University of Massachusetts (September 1974).

4. R. Baisey and L. Lieberman, "Computer Description of Real Outdoor Scenes," Proceedings 2nd International Joint Conference on Pattern Recognition, p. 174 (August 1974).

5. R. Haralick, "Textured Features for Image Classification," IEEE SMC, p. 160 (November 1973).

6. A. Rosenfeld and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis," IEEETC, p. 562 (May 1971).

7. F. Tomita, M. Yachida, and S. Tsuji, "Detection of Homogeneous Regions by Structured Analysis," Proceedings Third International Joint Conference on Artificial Intelligence, p. 564 (August 1973).

8. J. L. Muerle and D. C. Allen, "Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene," in Pictorial Pattern Recognition, pp. 3-13, Cheng et al., eds. (Thompson Press, Washington, D.C., 1968).

9. H. Blum, "A Tranformation for Extracting New Descriptors of Shape," in Models for the Perception of Speech and Visual Form, Wathen-Dunn, ed., pp. 362-380 (MIT Press, Cambridge 1967).

10. C. T. Zahn, "A Formal Description of Two Dimensional Patterns," Proceedings First International Joint Conference on Artificial Intelligence, Walker and Norton, eds., Washington, D.C., pp. 621-628 (1969).

11. H. Barrow and R. Popplestone, "Relational Descriptions in Picture Processing," in Machine Intelligence, Meltzer and Michie, eds., Vol. 6, pp. 377-396 (Edinburgh University Press, 1971).

12. C. T. Zahn and R. Z. Roskies, "Fourier Descriptors for Plane Closed Curves," IEEE Trans. on Computers, Vol. C-21, pp. 269-281 (1972).

13. H. Freeman, "Boundary Encoding and Processing," in Picture Processing and Psychopictorics, Lipkin and Rosenfeld eds., (Academic Press, pp. 241-306, 1970).

14. Fischler, "Machine Perception and Description of Pictorial Data," Proceedings First International Joint Conference on Artificial Intelligence, Washington, D.C., p. 629 (May 1969).

15. R. Narasimhan, "Picture Languages," in Picture Language Machines, S. Kaneff, ed., p. 1 (Academic Press, N.Y., 1970).

16. F. P. Preparata and S. R. Rae, "An Approach to Artificial Non-Symbolic Cognition," Information Science Vol. 4, pp. 65-86 (1972).

17. Y. Yakimovsky and J. A. Feldman, "A Semantics Based Decision Theoretic Region Analyzer," Proceedings 3IJCAI p. 580 (August 1973).

18. N. J. Nilsson et al., "Artificial Intelligence--Research and Applications," Progress Report to ARPA covering the period 9 October 1972 through 8 March 1974, Stanford Research Institute (April 1974).

19. C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," Artificial Intelligence, Vol. 1, No. 3, pp. 205-226 (1970).

20. A. Guzman, "Computer Recognition of Three-Dimensional Objects in a Visual Scene," Massachusetts Institute of Technology," MAC-TR-59 (December 1968).

21. S. L. Horowitz and T. Pavlidas, "Picture Segmentation by a Directed Split and Merge Procedure," Proceedings Second International Joint Conference on Pattern Recognition, Copenhagen, pp. 424-433 (August 1974).

22. R. Baisey, "Computer Identification of Visual Surfaces," Computer Graphics and Image Processing, pp. 118-130, Vol. 2, No. 2 (October 1973).

23. C. A. Harlow and A. Eisenpeis, "The Analysis of Radiographic Images," IEEETC, Vol. C-22, No. 7, p. 678 (July 1973).

24. E. Freuder, MIT AI Lab, personal communication.

25. T. D. Garvey, "Automatic Generation of Perceptual Strategies," Ph.D. Dissertation, Stanford University (forthcoming).

26. R. Duda and P. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, p. 75 (1973).

APPENDIX A1

COST and CONFIDENCE

This appendix defines the concepts of cost and reliability (or confidence) used in evaluating detectors, and derives their computational formulas.

Confidence

In the course of planning a strategy, or examining a picture interactively, an important question frequently arises: "Given a particular set of attribute measurements on an item (a region or sample), what set of objects could it possibly belong to?" If the measurements have already been taken from some region (or sample), the answer to the question allows the system to "classify" the item as belonging to some element of a set of objects. If the system is considering generating some detector for taking the measurements, then the answer to the question allows an estimate of how well the detector should work.

In addition to knowing the set of possible objects which could provide the measurements, it is also important to know the relative likelihood of each object in the set. For example, if the system knows that the local (surface) orientation of a sample is horizontal, then it should know not only that it belongs to either

tabletop or floor, but also that it is more likely to belong to the floor since the floor (usually) takes up much more of the image than does the tabletop. If the system also measures the height of the sample to be 2 1/2 feet, then it should realize that the sample has to belong to tabletop, since that is the only object which has both properties. Finally, if the measurements span a wide range of values, then several objects' attribute ranges may overlap the measured range. In this case, the system should take into account the amount of overlap.

We can formulate a set of requirements for a system which answers our original question. The system should account for a priori probabilities of the item belonging to an object, should be able to handle combinations of attribute measurements, and should be able to measure and use the degree of attribute range overlap.

The program that satisfies these requirements, CONF, measures the confidence that a sample belongs to an object, based on a set of detector outcomes. A recursive Bayes procedure is used in the computations [26]. Attribute measurements are taken as independent events which are linked together through object descriptions. That is, if no object descriptions were available, the measurements would be truly probabalistically independent events. However, having object descriptions allows the system to compute dependent probabilities.

Before beginning the discussion of the procedure, some notation is required. The set of outcomes of the application of a set of detectors, $\{D_n, D_{n-1}, \ldots, D_1\}$ will be written, D^n. A detector, D_k,

Since the outcome of a detector is affected only by the objects it is conditioned on, and not by the outcomes of other detectors, $P(D_n|O_i, D^{n-1})$ is reduced to $P(D_n|O_i)$ and the expression becomes:

$$P(O_i|D^n) = \frac{P(D_n|O_i) \times P(O_i|D^{n-1})}{P(D_n|D^{n-1})} \qquad (2)$$

This is the usual way of writing the recursive Bayes expression. $P(D_n|O_i)$ is the probability that a random sample from object, O_i, will have outcome D_n when the detector is applied. This probability is computed from stored data by summing the samples that would produce the given outcome, and dividing by the total number of samples measured. The data used for this step come from characterizations generated by the system from examples of the object.

The term, $P(O_i|D^{n-1})$ is computed recursively, terminating with $P(O_i|D_1)$ which is expanded in the normal way,

$$P(O_i|D_1) = \frac{P(D_1|O_i) \times P(O_i)}{P(D_1)} \qquad (3)$$

$P(O_i)$ is the a priori likelihood of a randomly selected sample belonging to O_i. This is usually computed by comparing the expected projected two-dimensional area of the object with the area of the image. $P(O_i)$ also reflects the set of objects expected to be present. By setting $P(O_i)$ to zero, the object is effectively eliminated from consideration.

To compute $P(D_1)$, $P(D_1|O_j)$ is summed over all objects.

$$P(D_1) = \sum_{O_j} P(D_1|O_j) \times P(O_j) \qquad (4)$$

Now, with the derivation of $P(O_i|D_1)$, and therefore the derivation of $P(O_i|D^{n-1})$, the final details for the original computation of $P(O_i|D^n)$ can be completed.

The last term to be expanded is $P(D_n|D^{n-1})$. This is where the detector outcomes are linked through the object descriptions. As in Equation 4, the term will be expanded over all objects.

$$P(D_n|D^{n-1}) = \sum_{O_j} P(D_n|D^{n-1}, O_j) \times P(O_j|D^{n-1})$$

As mentioned previously, the outcome of a detector depends on objects, not on other detectors except insofar as they select objects. Therefore, the term $P(D_n|D^{n-1}, O_j)$ is reduced to $P(D_n|O_j)$, and the expression is rewritten as

$$P(D_n|D^{n-1}) = \sum_{O_j} P(D_n|O_j) \times P(O_j|D^{n-1}) \qquad (5)$$

This expression is Equation 4, conditioned on the remaining set of detectors, D^{n-1}.

Several points need to be emphasized. It is important to be able to limit the set of objects under consideration. Limiting this set allows the system to reduce its window into the scene, from a full image, to a selected subimage. This reduction typically comes

about when the system locates some object and then looks in the immediate vicinity for other objects. The initial object then provides a new window into the scene, usually with an appreciably smaller subset of objects that need to be considered.

Another important point is that the derivations use the outcomes of the set of detectors, and do not depend on the value of the outcomes. That is, the computations do not require that all the detectors accept the sample, but only that there is some outcome. This fact allows for the possibility of generating decision trees where one branch of the tree is taken for an acceptance, and the other for a rejection. Although the system does not currently generate decision trees (but only conjunctions that require that all outcomes are acceptances), there are no theoretical barriers.

COST

In this discussion, cost will be the anticipated cost (measured in milliseconds of CPU time) of applying a detector to a sample (regions are not applicable here). In the discussion of confidence, the order of application of detectors was immaterial, since all outcomes were needed. In this discussion, detectors will be limited to composite detectors, i.e., conjunctions of simple detectors applied sequentially. The cost of application of a composite detector to an image is highly order dependent, since a rejection by one detector in a sequence implies that the remaining detectors in the sequence need not be applied.

Most of the notation required here was defined in the preceding section. Let $C_s(D^n)$ be the per sample cost of applying detector sequence, D^n, and let $C_s(D_i)$ be the per sample cost of the single detector, D_i. The single detector costs are measured previously by the system.

The cost of application of the detector sequence, D^n, is

$$C_s(D^n) = C_s(D_1) + C_s(D_2) \times P(D_1) + C_s(D_3) \times P(D_1, D_2) + \ldots$$

$$= \sum_{i=1}^{n} C_s(D_i) \times P(D^{i-1}) \quad . \tag{6}$$

D_1 is always applied. A fraction of the samples tested with D_1 (i.e., those accepted) will also be tested with D_2. The percentage of samples passed by D_1 and D_2 will also be tested by D_3, and so on.

The cost of application of a detector to the image (or to a window) is the per sample cost of the detector times the number of samples to be tested. Since the number of samples to be tested also depends on the detector, that number is derived here. It is assumed that a certain given number of samples on the target object should be accepted by the detector. This number, preset by the user, is N_D. From N_D and some object data, it is possible to compute a sampling density, δ, which will provide a sufficient number of samples on the object, such that N_D should be accepted by the detector. N_T is the total number of samples that should fall on the object, given a sampling density, δ, and the expected object size, S(O).

$$N_T = \delta \times S(O) \tag{7}$$

But since only the fraction, $P(D^n|O)$, of any random sampling of O can be expected to be accepted by D^n,

$$N_D = N_T \times P(D^n|O) \quad . \tag{8}$$

Therefore,

$$N_D = \delta \times S(O) \times P(D^n|O) \quad , \tag{9}$$

and

$$\frac{N_D}{S(O) \times P(D^n|O)}$$

Since the system does not retain the correlated data necessary to determine $P(D^n|O)$, it instead uses the minimum of these set of probabilities, $\{P(D_1|O), \ldots, P(D_n|O)\}$.

With δ, the total number of samples, N_W, to be checked in a window, W, is the window size times δ, or

$$N_W = \delta \times S(W) \quad . \tag{10}$$

This gives a total cost of

$$C(D^n) = N_W \times C_s(D^n) \quad .$$

$$\frac{N_D}{P(D^n|O)} \times \frac{S(W)}{S(O)} \times C_s(D^n) \tag{11}$$